UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

Aplicación para registrar ensayos no destructivos sobre piezas de aviación

Autor/es

JEAN FRANCO TIRAVANTTI BILBAO

Director/es

EDUARDO SÁENZ DE CABEZÓN IRIGARAY

Facultad

Facultad de Ciencia y Tecnología

Titulación

Grado en Ingeniería Informática

Departamento

MATEMÁTICAS Y COMPUTACIÓN

Curso académico

2019-20

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE GRADO

Título
Aplicación para registrar ensayos no destructivos en piezas de aviación
Autor
JEAN FRANCO TIRAVANTTI BILBAO
Tutor
EDUARDO SÁENZ DE CABEZÓN IRIGARAY
Facultad
FACULTAD DE CIENCIAS Y TECNOLOGÍA
Titulación
GRADO EN INGENIERÍA INFORMÁTICA
Curso Académico
2019-2020

Resumen

El presente Trabajo Fin de Grado tiene el objetivo de crear una aplicación de escritorio que permita registrar los ensayos no destructivos que se aplican a las piezas de aviación, habiendo que registrar 3 procesos principales: Secado del bastidor, Inspección Por Líquidos Penetrantes y Dureza y Conductividad.

Una vez desarrollada la interfaz de la aplicación, además de habilitar las distintas funcionalidades para llevar a cabo los distintos procesos, implementaremos el uso de tarjetas NFC[1] para facilitar el sellado a los operarios de la fábrica. De esta manera, solo aquellos operarios que dispongan de una tarjeta con cierto nivel de privilegio podrán sellar bastidores.

Como resultado final de este trabajo, tenemos un software para escritorio el cual nos va a permitir monitorizar los bastidores que están pendientes de realizar algunos de los procesos, las distintas pantallas para registrar los resultados obtenidos de las pruebas realizadas a las piezas de los bastidores y sellar los bastidores haciendo uso de las tarjetas NFC.

Abstract

The aim of this Bachelor Thesis is to make a desktop application that allows registering non-destructive tests used on aviation parts, having to register 3 main processes: Drying of the frame, Inspection for Penetrating Liquids and Hardness and Conductivity.

Once the application interface was developed, in addition to enabling the different functionalities to carry out the process of the different processes, we implemented the use of NFC cards to facilitate sealing to factory operators. In this way, only those operators who have a card with a certain level of privilege can seal racks.

As outcome, we have desktop software that will allow us to monitor the racks that are pending some of the processes, the different screens to register the results of the tests carried out on the parts of the racks and seal the racks using NFC cards.

[1] Near-field communication (NFC) o comunicación de campo cercano es una tecnología de comunicación inalámbrica, de corto alcance y alta frecuencia que permite el intercambio de datos entre dispositivos.

UNIVERSIDAD
DE LA RIOJA

Facultad de Ciencia y Tecnología

TRABAJO DE FIN DE GRADO

Grado en Ingeniería Informática

Aplicación para registrar ensayos no destructivos en piezas de aviación

Realizado por:
Jean Franco Tiravantti Bilbao

Tutelado por:

Eduardo Sáenz de Cabezón Irigaray

Logroño, Junio 2020

Índice

1. Introducción

1.1. Antecedentes y estudio de viabilidad

Mecanizaciones Aeronáuticas, SA (MASA) es una empresa dedicada a la fabricación de piezas mediante máquinas de control numérico, tratamiento y montaje de conjuntos metálicos para la industria aeronáutica. Este proyecto me ha sido propuesto por la empresa, en concreto por el Departamento de Mejora.
En el Departamento de Mejora estamos trabajando continuamente en progresar y mejorar el sistema de gestión de la empresa con el objetivo de concebir hoy las soluciones del mañana, tanto en máquinas como en procedimientos.

Actualmente en la fábrica los operarios utilizan una aplicación de escritorio para registrar los datos obtenidos de una serie de procesos que se llevan a cabo en las distintas piezas de aviación, dicha aplicación está desarrollada en Visual Basic con Access, un lenguaje de programación que hoy en día cada vez se usa menos.
Cada vez resulta más complicado añadir nuevas funcionalidades y mejoras a la versión de la aplicación actual, además de la incompatibilidad con el resto de los sistemas. Por ello surgió la idea de desarrollar una nueva aplicación que dé solución a estos problemas.

El objetivo del proyecto es la creación de una versión mejorada de la anterior existente desarrollada en Access, para ello se ha decidido utilizar el lenguaje de programación C# y .NET Framework ya que es un lenguaje elegante, con seguridad de tipos y orientado a objetos. Haremos uso del entorno de desarrollo "Visual Studio", un entorno potente para la realización de este proyecto.

Por otro lado, también será necesario migrar las tablas usadas en la Base de Datos desarrollada en Access a la nueva Base de Datos existente en Microsoft SQL Server. Únicamente será necesario crear aquellas tablas que utiliza la aplicación actual en Access ya que la base de datos contiene mucha más información de la que haremos uso, por lo que habrá que crear solo aquellas tablas que no existan en la nueva Base de Datos en SQL Server.
Esta base de datos en Access sigue activa debido a que aún existen aplicaciones hechas en Visual Basic con Access que continúan haciendo uso de ella.
Además, también haremos uso de determinadas tablas de la Base de Datos en SQL Server para la realización de nuestra aplicación.
En definitiva, la migración de las tablas a la nueva base de datos en SQL Server supondrá una mejora del rendimiento y escalabilidad, seguridad mejorada, más usuarios simultáneos, mayor disponibilidad, entre otras mejoras.

El sistema permitirá a los operarios de la fábrica registrar y gestionar los resultados obtenidos de las pruebas realizadas durante los ensayos no destructivos en piezas de aviación de una manera más segura y eficaz, además de disponer de una interfaz más sencilla e intuitiva.

Por lo tanto, en el Departamento de Mejora de MASA se ha aprobado el desarrollo de una nueva aplicación que mejore a la anterior suponiendo un avance en el procedimiento y que facilite el trabajo a los operarios.

1.2. Objetivo

Teniendo en cuenta lo anterior, la idea del proyecto se centrará en el desarrollo de una aplicación de escritorio que permita recoger los resultados obtenidos tras la ejecución de 3 procesos:

- Secado del bastidor
- Inspección por Líquidos Penetrantes
- Dureza y Conductividad

Para la realización de este proyecto tendré que aprender conceptos básicos de los procedimientos que se van a llevar a cabo en estos procesos. Para ello cuento con la ayuda de mis compañeros de departamento, pero sobre todo de los operarios de fábrica quienes conocen mejor estos procesos y quienes serán los usuarios finales de esta aplicación. Por lo tanto, tendremos que coordinarnos para conseguir un resultado lo más próximo a lo esperado.

2. Definición del sistema

2.1. Análisis

Este proyecto consistirá en la creación de una aplicación de escritorio para Windows mediante el cual los operarios de MASA podrán gestionar y registrar los datos obtenidos de los ensayos no destructivos de piezas de aviación.

El sistema ha de permitir realizar 3 procesos principales:

- Secado del bastidor: las piezas son introducidas en una estufa a una determinada temperatura y durante un periodo de tiempo fijado. Una vez finalizado el tiempo son retiradas las piezas y se registran los datos obtenidos del proceso.
  Para este proceso recogeremos los siguientes datos:

Secado Bastidor		Valores Teóricos
Secado	Temperatura (ºC)	60-70 ºC
	Tiempo (Min)	> 45 Min

- Inspección por Líquidos Penetrantes: durante este proceso se aplica un líquido revelador que permite identificar posibles defectos en el material de piezas aeronáuticas. Se tienen en cuenta factores como el tiempo de revelado, temperatura de la sala, se registran las piezas que hayan presentado problemas.
  Para este proceso recogeremos los siguientes datos:

Inspección por Líquidos Penetrantes		Valores Teóricos
Temperatura (ºC)		11-38 ºC
Tiempo (Min)	Penetrante	10-60 Min
	Revelado	20-60 Min

- Dureza y Conductividad: durante este proceso se aplican a las piezas métodos para medir su dureza usando un durómetro y para medir la conductividad eléctrica se utiliza un medidor electrónico.

	Conductividad	Dureza	Valores Teóricos
Criterios	%IACS	MS/m	Rockwell[1]
A	2.0	1.16	------
B	3.0	1.74	3.5 HRb

El criterio A será utilizado cuando no se requiera inspección de dureza.
El criterio B será utilizado cuando sean requeridas simultáneamente inspecciones de dureza y conductividad.

Para el sellado[2] de cada bastidor implementaremos un método de escaneo de tarjetas NFC, ya que se disponen de lectores de tarjetas en los puestos de trabajo por lo que cada operario podrá hacer uso de su tarjeta para sellar, de esta forma además de registrar los datos también almacenaremos en la base de datos los bastidores que han sido sellados por cada operario.

Con este método agilizaremos el proceso de sellado ya que actualmente han de hacer un "login" en cada uno de ellos.

2.2. Alcance

El alcance de este proyecto consiste en crear una aplicación que permita realizar además de los 3 procesos mencionados anteriormente, visualizar los bastidores pendientes de trabajo, los bastidores que ya han sido sellados, es decir, obtener un historial de los distintos procesos. Por lo tanto, permitirá a los operarios tener en todo momento un acceso a este histórico. Además, también habrá que añadir la funcionalidad para trabajar con lectores y tarjetas NFC.

Los bastidores pasan por una serie de fases, cada fase es supervisada por un operario, una vez que finalice ese proceso y esté listo para la siguiente fase cada operario ha de sellar con su tarjeta. De esta forma se lleva un control de los bastidores en cada fase, y así sabemos que operario sello qué bastidores para posteriores controles. Si un bastidor no ha sido sellado no puede pasar a la siguiente fase y los bastidores solo pueden ser sellados por operarios autorizados, aquellos que dispongan de una tarjeta NFC.

En primer lugar, me centraré en explicar cómo se llevan a cabo los procesos en Grietas, así es como llamamos a este conjunto de procesos que se realizan durante esta sección de trabajo que son: Secado del Bastidor, Inspección Por Líquidos Penetrantes, Dureza y Conductividad.

[1] Máxima diferencia de dureza medida en los puntos de máxima y mínima conductividad obtenida
[2] Confirmación de que los datos obtenidos y el proceso han sido correctos

En cuanto al proceso de Secado, los bastidores con los que se trabaja en este proceso son los que han sido sellados en la anterior sección de trabajo llamada Baños, no entrare en detalles de que procesos se realizan en esa sección ya que no influye en nuestro sistema; una vez llegan estos bastidores a nuestro sistema pasan a un estado de pendientes, cabe señalar que solo nos llegan bastidores de tipo Decapado y Titanio. Todos los batidores que nos llegan de Baños son un conjunto de piezas de aviación que han de realizar obligatoriamente los procesos de Secado e Inspección Por Líquidos Penetrantes y el proceso de Dureza y Conductividad. Se realiza el proceso de Secado a los bastidores pendientes que van llegando de Baños, una vez son sellados en este proceso pueden pasar al siguiente, al proceso de Inspección Por Líquidos Penetrantes.

En el proceso de Inspección Por Líquidos Penetrantes puede ocurrir que algunas piezas no pasen la prueba y se requiera un retrabajo, pero únicamente para esa pieza o piezas en concreto por lo que será necesario habilitar la funcionalidad de añadir estas piezas a bastidores que aún no han realizado este proceso. Cuando se sella el bastidor, en este proceso, no pasa al siguiente, sino que finaliza en este último, en el proceso de Inspección Por Líquidos Penetrantes.

Finalmente, el proceso de Dureza y Conductividad, que se realiza a todas las piezas de aviación, no vienen en bastidores, sino que se crean lotes para las mismas, es decir, una vez finalizados los procesos previos se crean lotes de piezas de distintos bastidores, esto no afecta al proceso ya que las pruebas que se realizan se hacen pieza por pieza. Al crear un lote habrá que permitir en el sistema la funcionalidad de añadir las piezas que se quieren analizar en ese lote durante el proceso.

2.3. Requisitos funcionales

- El sistema controlará el acceso y lo permitirá solamente a usuarios autorizados a través de tarjetas NFC.
- El sistema recuperará y actualizará la base de datos automáticamente cada vez que se inicie la aplicación.
- El sistema permitirá únicamente la visualización del trabajo en curso e histórico a los usuarios no registrados.
- El sistema impedirá a los usuarios no registrados ingresar o aprobar información.
- El sistema permitirá a los usuarios autorizados el ingreso de datos tras el proceso de secado.
- El sistema permitirá a los usuarios autorizados el ingreso de datos obtenidos de las pruebas de inspección por penetrantes.
- El sistema permitirá a los usuarios autorizados el registro de datos de las pruebas de conductividad y dureza.
- El sistema permitirá a los usuarios autorizados la impresión de etiquetas con los datos registrados.
- El sistema permitirá a los usuarios autorizados aprobar, cambiar o actualizar los datos ingresados.
- El sistema validará los campos con la información ingresada de tal forma que solo aceptará aquellos datos que sean correctos en función de una serie de normas.

2.4. Requisitos no funcionales

- La aplicación se realizará en el lenguaje de programación C# y con el entorno de desarrollo de Visual Studio, ambas usadas en la empresa.
- La gestión de la información se realizará a través de Microsoft SQL Server.
- La aplicación debe proporcionar tiempos de respuestas rápidos.
- La interfaz de la aplicación debe ser intuitiva y sencilla de usar.
- La aplicación debe proporcionar seguridad al usuario, es decir, debe impedir la instalación de malware.

2.5. Selección de tecnologías

- Microsoft Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado disponible para Windows, Linux y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc., a lo cual hay que sumarle las nuevas capacidades en línea bajo Windows Azure. Visual Studio permite crear sitios y aplicaciones web, así como servicios web en cualquier entorno compatible con la plataforma .NET.

Características

• Desarrollo:
  • Aplicaciones y juegos para todos los dispositivos que ejecutan Windows.
  • Aplicaciones nativas o híbridas para Android, iOS y Windows.
  • Crear, administrar e implementar fácilmente aplicaciones de escala de nube en Azure.
  • Desarrollar aplicaciones web modernas con flexibilidad y eficaces herramientas de código abierto.
  • Permite escribir nuestras propias extensiones para Visual Studio.
  • Desarrollar e implementar bases de datos de SQL Server y Azure SQL.
• Tecnologías: C++, C#, Node.js, Python, .NET, JavaScript/TypeScript
• Depuración: Disponible para varios lenguajes en cualquier plataforma o ubicación (local, remota o en producción), además permite un control especifico e inspección flexible del estado.
• Pruebas: permite escribir, ejecutar y depurar pruebas unitarias en el lenguaje y el marco de pruebas que elegimos. También está disponible la creación de pruebas unitarias dinámicas.
• Colaboración: control de versiones distribuida o centralizado, flexible y ampliable, con extensiones (GIT o TFVC[1]).
• Extensión: Dispone de miles extensiones para personalizar nuestro IDE.

[1] Control de versiones de Team Foundation

- Microsoft SQL Server

Microsoft SQL Server es un sistema de gestión de bases de datos relacional de código abierto (RDBMS[1]) creado por Microsoft. El lenguaje de consulta y de desarrollo (por línea de comandos o mediante la interfaz gráfica de Management Studio) que utiliza es Transact-SQL[2].

Ventajas	Desventajas
• Facilidad de soporte de transacciones. • Alta escalabilidad, estabilidad y seguridad. • Soporta procedimientos almacenados. • Puede incluir un entorno gráfico de administración que permita el uso de comandos DDL[3] Y DML[4] gráficamente. • Modo cliente-servidor, los datos se alojan en el servidor y las terminales o clientes de la red solo han de acceder a la información. • Permite administrar información de otros servidores de datos	• La principal desventaja es que requiere una gran cantidad de memoria RAM para la instalación y uso de software. • Requiere de licencia. • Disponible solo para Windows. • Limitación de conexiones simultaneas para las páginas.

- DevExpress

DevExpress es una de las más completas suites de componentes de UI para el desarrollo en todas las plataformas de .NET como Windows Forms, ASP.NET, MVC, Silverlight y Windows 8 XAML.
Componentes que incluye:

- Tablas
- Calendario
- Editor HTML
- Hojas de cálculo
- Editores de datos
- Gráficas

[1] Sistema de gestión de bases de datos relaciones
[2] Es una extensión al SQL de Microsoft y Sybase.
[3] Data Definition Language (Lenguaje de definición de datos)
[4] Data Manipulation Language (Lenguaje de manipulación de datos)

Ventajas	Desventajas
• Cuenta con controles para todas las plataformas de Microsoft Windows. • Posee más de 70 controles mediante los cuales se pueden diseñar aplicaciones de alta complejidad. • La creación de los componentes es semi-automática, DevExpress se encarga de realizar todo el código necesario para la visualización y llenado de los componentes según la plataforma utilizada. • Permite llenar de manera sencilla cada uno de los componentes con información traída de una conexión de base de datos. • Mejora el rendimiento de las aplicaciones al optimizar el código de llenado de las vistas. • Funciona en cualquier explorador (Aplicaciones Web). • El tiempo de desarrollo se reduce considerablemente al no tener que teclear todo el código. • Incluye plantillas predefinidas para varios tipos de aplicaciones. • Incluye la herramienta Theme Builder, la cual permite editar el estilo de los controles y genera automáticamente la hoja de estilos de la aplicación siguiendo el patrón elegido. • La documentación que incluye tanto en Visual Studio (descripciones de métodos y parámetros) como externa (documentación de clases y ejemplos) es realmente extensa y útil.	• Solo funciona para plataformas de Microsoft. • Existen algunas limitantes al incluir los componentes, ya que estos ya tienen características y comportamientos definidos. • Requiere licencia.

- Lectores y tarjetas NFC:

  En la empresa se emplean lectores y tarjetas NFC en los puestos de trabajo de los operarios para agilizar distintos procesos siendo estos inicios de sesión, sellado de bastidores, cajas, es decir todo proceso aquel que requiera de la autorización o validación de un operario que disponga de estas tarjetas ya que solo están disponibles para los operarios de cierto rango.

  Para acoplar este funcionamiento en mi programa emplearé la librería winscard.dll la cual dispone de funciones que permiten conectar el PC con tarjetas inteligentes (PC/SC), es decir, la aplicación por medio de estas funciones se comunicará con el lector y la tarjeta para identificar el UID[1] de la tarjeta.

[1] El chip de la tarjeta contiene un número de serie único para fines de identificación, esto se llama UID. Entonces podemos identificar la etiqueta el uno del otro.

Lector NFC - ACR122u

Tarjetas NFC – Mifare Classic 1K

3. Planificación

3.1. Definición, distribución y estimación temporal de tareas

EDT

A continuación, en la figura 2.1, podemos ver la estructura de descomposición de tareas (EDT), en la cual definimos los diferentes paquetes de trabajo en que está dividido el proyecto.

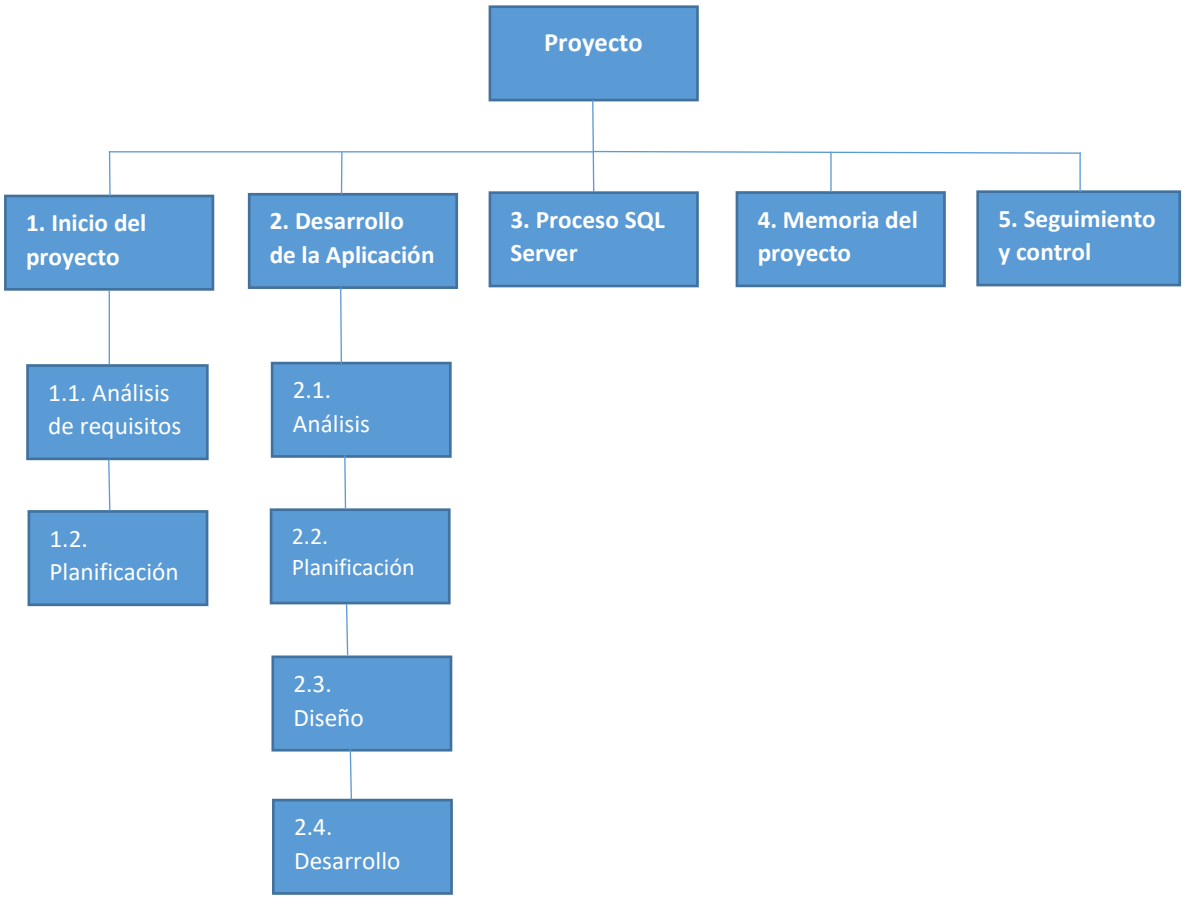


Figura 1 – Estructura de descomposición de tareas

La siguiente tabla muestra la división en tareas del proyecto, con su fecha de inicio y fin, además de las horas totales que estimo dedicar a cada una de ellas.

TAREA	INICIO	FIN	HORAS TOTALES
1. Definición del sistema	**3 de febrero**	**4 de febrero**	**15 horas**
1.1. Análisis	3 de febrero	5 de febrero	3 horas
1.2. Alcance	4 de febrero	5 de febrero	3 horas
1.3. Requisitos del sistema	5 de febrero	8 de febrero	5 horas
1.4. Selección de tecnologías	7 de febrero	10 de febrero	4 horas
2. Planificación	**10 de febrero**	**17 de febrero**	**13 horas**
2.1. Planificación temporal	10 de febrero	14 de febrero	8 horas
2.2. Cronograma	13 de febrero	15 de febrero	3 horas
2.3. Definición de metodologías	15 de febrero	17 de febrero	2 horas
3. Diseño e Implementación	**17 de febrero**	**8 de abril**	**180 horas**
3.1. Planificación iteraciones	17 de febrero	21 de febrero	5 horas
3.2. Diseño	24 de febrero	8 de abril	10 horas
3.3. Desarrollo de la aplicación	24 de febrero	8 de abril	165 horas
4. Pruebas	**28 de febrero**	**8 de abril**	**25 horas**
4.1. Pruebas SW	28 de febrero	8 de abril	20 horas
4.2. Pruebas tarjetas NFC	20 de marzo	8 de abril	5 horas
5. Seguimiento y control	**3 de febrero**	**3 de junio**	**35 horas**
5.1. Duración real de las tareas	3 de febrero	3 de junio	10 horas
5.2. Cronograma real	5 de febrero	3 de junio	10 horas
5.3. Justificación de desviaciones	17 de febrero	3 de junio	12 horas
5.4. Reuniones	3 de febrero	3 de junio	5 horas
6. Memoria y anexos	**5 de febrero**	**3 de junio**	**40 horas**

Figura 1.1 - Descripción de las tareas de la EDT

Figura 2 - Proporción de dedicación estimada

3.2. Cronograma

También he elaborado un diagrama de Gantt para representar el tiempo de dedicación previsto a cada una de las tareas del proyecto a lo largo de los meses de Febrero-Marzo-Abril-Mayo.

Cronograma																				
Paquetes de trabajo		Febrero																		
		3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1.1.	Análisis	■	■	■																	
1.2.	Alcance		■	■																	
1.3.	Requisitos del sistema			■	■	■	■														
1.4.	Selección de tecnologías					■	■		■												
2.1.	Planificación temporal								■	■	■	■	■								
2.2.	Cronograma											■	■	■							
2.3.	Definición de metodologías													■		■					
3.1.	Planificación iteraciones															■	■	■	■	■	
5.3.	Justificación desviaciones															■	■	■	■	■	■
5.4.	Reuniones	■	■	■					■	■	■					■	■	■			
6.	Memoria y anexos			■	■	■			■	■	■	■	■			■	■	■	■	■	

Cronograma																				
Paquetes de trabajo		Febrero							Marzo											
		23	24	25	26	27	28	29	1	2	3	4	5	6	7	8	9	10	11	12	13
3.2.	Diseño y Diagramas UML																				
3.3.	Desarrollo del sistema																				
4.1.	Pruebas Software																				
5.3.	Justificación desviaciones																				
5.4.	Reuniones																				
6.	Memoria y anexos																				

Cronograma																					
Paquetes de trabajo		Marzo																		Abril	
		14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2
3.2.	Diseño y Diagramas UML																				
3.3.	Desarrollo del sistema																				
4.1.	Pruebas software																				
4.2.	Pruebas Tarjetas NFC																				
5.3.	Justificación desviaciones																				
5.4.	Reuniones																				
6.	Memoria y anexos																				

Cronograma																					
Paquetes de trabajo		Abril																			
		3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
3.3.	Desarrollo del sistema																				
5.1.	Pruebas Software																				
5.3.	Justificación desviaciones																				
5.4.	Reuniones																				
6.	Memoria y anexos																				

Cronograma																					
Paquetes de trabajo		Abril								Mayo											
		23	24	25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12
5.1.	Pruebas Software																				
5.2.	Pruebas tarjetas NFC																				
5.3.	Justificación desviaciones																				
5.4.	Reuniones																				
6.	Memoria y anexos																				

Cronograma																					
Paquetes de trabajo		Mayo																			
		13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1
5.3.	Justificación desviaciones																				
5.4.	Reuniones																				
6.	Memoria y anexos																				

Cronograma																					
Paquetes de trabajo		Junio																			
		2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
5.3.	Justificación desviaciones																				
5.4.	Reuniones																				
6.	Memoria y anexos																				

3.3. Definición de metodologías

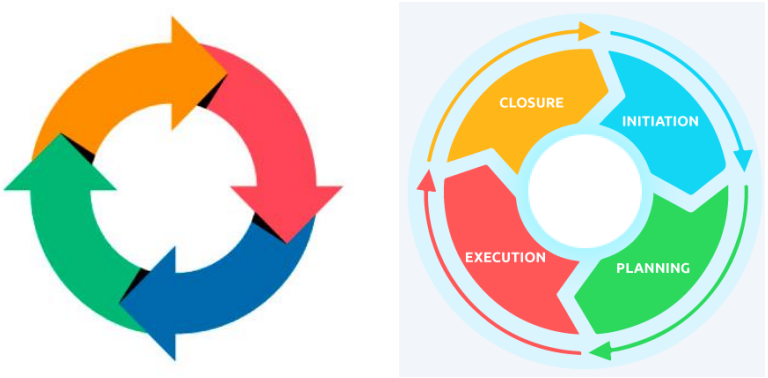


Figura 3. Fases del proyecto

3.3.1. Metodología de gestión del proyecto

La metodología elegida es una metodología iterativa e incremental. En un desarrollo iterativo e incremental el proyecto se planifica en diversos bloques o etapas temporales llamados iteraciones o sprints.

El desarrollo de esta metodología consistirá en dividir el proyecto en varios sprints, en los cuales iré desarrollando e implementado las diferentes funcionalidades del sistema.

Las principales razones del uso de un ciclo de desarrollo iterativo e incremental para la ejecución de este proyecto son:

- Sistema modular. Las características del sistema permiten desarrollar una base funcional mínima y sobre ella ir incrementando las funcionalidades o modificando el comportamiento o apariencia de la ya implementadas.
- Entregas frecuentes y continuas al cliente (en este caso mi tutor de prácticas) de los módulos terminados, de forma que puede disponer

de una funcionalidad básica en un tiempo mínimo y a partir de ahí un incremento y mejora continua del sistema.

- Previsible inestabilidad de requisitos.
- Es posible que el sistema incorpore más funcionalidades de las inicialmente identificadas.
- Es posible que durante la ejecución del proyecto se altere el orden en el que se desean recibir los módulos o historias de usuario terminadas.
- Desarrollo de software de calidad, debido a las pruebas y la valoración de cada una de las iteraciones (revisión y retrospectiva).

Habrá únicamente 2 roles: mi tutor de prácticas y yo.
La duración de cada sprint será de 1 semana.
En cada uno de los sprints se llevará a cabo:

- Planificación: Análisis y diseño
- Ejecución: Desarrollo del software
- Revisión y retrospectiva

Al finalizar cada sprint se presentará la funcionalidad desarrollada al cliente, en este caso mi tutor de prácticas, de forma que podrá decidir si cambiar alguno de los aspectos desarrollados durante ese sprint.
De esta manera conseguimos que la aplicación se adapte perfectamente a las necesidades de la empresa.

3.3.2. Metodología de gestión de versiones

Para el sistema de control de versiones del código, utilizaré Git. De esta forma podré gestionar todo el código de manera remota, haciendo uso del servicio en la nube gratuito GitHub.
Las principales razones por las que se ha decidido hacer uso de GitHub son:

- Permite hacer un seguimiento de los cambios en las diferentes versiones, facilitando la evaluación y el seguimiento del proyecto.
- Accesibilidad. GitHub nos permite acceder en cualquier momento a nuestro código.
- Compatibilidad. GitHub es una plataforma web, por tanto, es independiente del sistema operativo que usemos.
- Es gratuito.
- Experiencia con este sistema de control de versiones. Ya he trabajado anteriormente con GitHub en otros proyectos.

4. Diseño e Implementación

4.1. Planificación iteraciones

Como ya he mencionado anteriormente en la definición de la metodología, cada sprint durará 1 semana, hemos decidido emplear este tiempo en cada sprint ya que nos va a permitir avanzar de manera continua y efectuar cambios durante el proceso.

La realización de todos los sprints durará aproximadamente 5 semanas.
En el siguiente cronograma represento las semanas de los meses de febrero y marzo que emplearé para cada uno de los sprints.

Cronograma																				
Sprint		Febrero						Marzo												
		24	25	26	27	28	29	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1.	Historial y Trabajo en Curso																				
2.	Proceso Secado																				
3.	Proceso Inspección por Líquidos Penetrantes																				

Cronograma																				
Paquetes de trabajo		Marzo																Abril		
		15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3
4.	Proceso Dureza y Conductividad																				
5.	Tarjetas NFC																				

Voy a realizar un análisis inicial sobre las tareas que hemos planificado realizar en cada sprint.

1º Sprint: Historial y Trabajo en curso

Este sprint lo vamos a dedicar a la creación de la ventana principal del programa, que nos servirá de base para ir implementando las diferentes funcionalidades de nuestra aplicación.

La primera funcionalidad para crear serán los históricos de cada proceso (Secado, Inspección por Líquidos Penetrantes y Dureza Y Conductividad), para cada una de estas funcionalidades se dispondrá de un botón en la ventana principal de la aplicación.

La segunda funcionalidad que añadiremos durante esta iteración será la visualización del trabajo que se encuentra en curso, es decir, los bastidores que están pendientes de verificación y sellado por los operarios. Además, por cada bastidor habrá un botón que permita acceder directamente al proceso correspondiente en cada caso. Vamos a crear de momento esta funcionalidad únicamente para los procesos de Secado e Inspección por Líquidos Penetrantes, más adelante crearemos la de Dureza y Conductividad.

2º Sprint: Proceso Secado: Decapado (Aluminio) y Titanio

En esta iteración crearemos las ventanas para el registro de datos del proceso, según el tipo de bastidor se mostrará una ventana u otra ya que para cada tipo

interesa almacenar datos concretos por lo que resulta necesario crear una ventana por tipo, una para Decapado y otra para Titanio.

Además, en este sprint añadiremos funcionalidad a los botones de la ventana de Trabajo en Curso creados en el sprint anterior, para que al hacer clic nos dirija a la ventana del bastidor seleccionado y poder editar sus datos.

Notas: Este proceso es específico ya que únicamente se realiza a bastidores de tipo Titanio o Decapado, existe un proceso anterior a este llamado “Baños” por el cual pasan todos los bastidores, pero este proceso solo se aplica a los bastidores que sean de los tipos mencionados antes.

3º Sprint: Proceso Inspección por Penetrantes

En este sprint vamos a crear la ventana de registro de datos para este proceso. A diferencia con respecto a la fase de Secado, solo será necesario crear una ventana para ambos tipos de bastidor ya que los datos que registraremos son comunes en ambos.

Además de recoger datos correspondientes al bastidor, en este proceso es necesario registrar información de las piezas que van en cada bastidor, por lo que habrá que permitir a los operarios ver y editar los datos de cada pieza.

Nota: A este proceso llegan los bastidores que han sido sellados por los operarios en la fase anterior (Secado del bastidor), se realiza la Inspección por Líquidos Penetrantes a ambos tipos de bastidores (Titanio y Aluminio).

4º Sprint: Proceso Dureza y Conductividad

En este sprint realizaremos la ventana de registro de datos para el proceso de Dureza y Conductividad. Para llevar a cabo este proceso debemos permitir a los operarios crear lotes de piezas, a diferencia con los otros procesos aquí no hay bastidores sino lotes los cuales pueden estar formados por piezas de bastidores diferentes. Además, también tenemos que permitir a los operarios registrar información de las piezas de cada lote.

Otra funcionalidad que desarrollaremos en esta iteración será la visualización de los lotes pendientes, lotes que han sido creados pero que aún no han sido sellados. De esta manera permitiremos a los operarios retomar el trabajo inacabado.

Nota: El proceso de Dureza y Conductividad es independiente con respecto a los demás procesos, por lo que se puede realizar en cualquier momento, no es necesario que se haya realizado alguna de las fases anteriores para poderse llevar a cabo.

5º Sprint: Implementación lectura tarjetas NFC

En esta iteración incorporaremos a la aplicación la lectura de tarjetas NFC por medio de lectores NFC para el sellado tanto de los bastidores en los procesos de Secado e Inspección por Líquidos Penetrantes y el sellado de los lotes en el proceso de Dureza y Conductividad. De esta manera mantendremos un control de autorización para la confirmación de las acciones.

4.2. Diseño

4.2.1. Análisis de requisitos

A continuación, se muestran las necesidades del sistema software. Para facilitar el proceso y representar de manera más clara, vamos a utilizar las técnicas de análisis y diseño de software empleadas en la asignatura de Diseño Tecnológico de Sistemas de Información que se basan en el lenguaje de modelado UML, un lenguaje gráfico que nos permite visualizar, especificar y documentar nuestro sistema.

4.2.2. Diagramas de Casos de Uso

A continuación, se representa el análisis de comportamiento del sistema en forma de casos de uso, identificando los actores y sus acciones.
En nuestro sistema existen dos actores principales: usuario y operario.

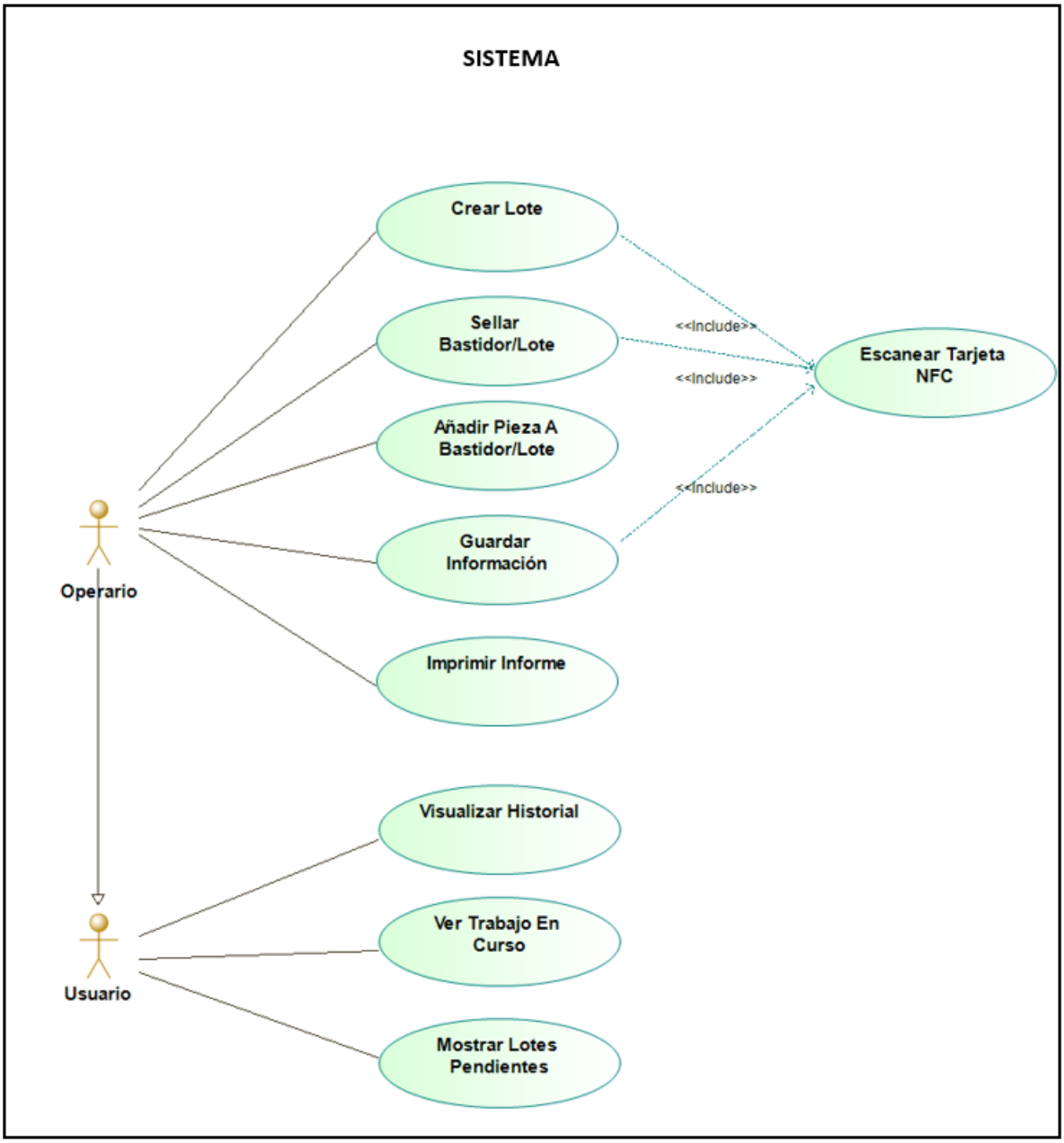


Figura 4 – Diagrama de casos de uso

Primero voy a definir a los Actores:

- Usuario: Cualquier persona que tenga acceso a la aplicación pudiendo ser este un operario ayudante de la fábrica, algún empleado de administración que necesite obtener información de los históricos o de los bastidores pendientes (Trabajo En Curso y Lotes Pendientes)

- Operario: Personal de la fábrica que dispone de una tarjeta NFC única y personal, de esta manera aseguramos que la información proporcionada sobre los bastidores ha sido comprobada por el personal profesional de la fábrica.

Con la finalidad de obtener mayor detalle, a continuación, especificaremos brevemente cada caso de uso:

Caso de uso	Visualizar Historial
Actores	Usuario, Operario
Resumen	Permite visualizar un resumen de los históricos de los 3 procesos posibles: • Secado (Decapado y Titanio) • Inspección por Líquidos Penetrantes • Dureza y Conductividad
Precondición	Establecer conexión con la BD
Postcondición	Muestra el historial de los 3 procesos

Caso de uso	Ver Trabajo En Curso
Actores	Usuario, Operario
Resumen	Permite visualizar los bastidores pendientes de sellado de los procesos: • Secado (Decapado y Titanio) • Inspección por Líquidos Penetrantes
Precondición	Establecer conexión con la BD
Postcondición	Muestra los bastidores pendientes de sellado de los procesos mencionados

Caso de uso	Mostrar Lotes Pendientes
Actores	Usuario, Operario
Resumen	Permite visualizar los lotes pendientes de sellado del proceso de Dureza y Conductividad
Precondición	Establecer conexión con la BD
Postcondición	Muestra los lotes pendientes de sellado del proceso mencionado

Caso de uso	Imprimir Informe
Actores	Usuario, Operario
Resumen	Permite obtener un PDF de resumen del bastidor seleccionado para el proceso de Inspección por Líquidos Penetrantes
Postcondición	Muestra un PDF de resumen del proceso

Caso de uso	Guardar cambios
Actores	Operario
Resumen	Permite guardar el estado del bastidor que se está actualizando, de esta manera se puede salir y continuar con el proceso más adelante.
Precondición	- Establecer conexión con la BD - Necesario pasar tarjeta NFC para confirmar acción
Postcondición	Se actualiza en la BD los datos modificados del bastidor seleccionado

Caso de uso	Sellar Bastidor/Lote
Actores	Operario
Resumen	Permite confirmar los datos proporcionados para bastidor o lote sellándolo
Precondición	- Establecer conexión con la BD - Necesario pasar tarjeta NFC para confirmar acción
Postcondición	Se actualiza en la BD con los datos modificados del bastidor o lote seleccionado y se eliminada de los bastidores o lotes pendientes

Caso de uso	Crear lote
Actores	Operario
Resumen	Permite crear un nuevo registro de lote para el proceso de Dureza y Conductividad
Precondición	- Solicitará que se guarden los cambios para crear el lote por que será necesario pasar tarjeta NFC por el lector para confirmar el registro - Establecer conexión con la BD - Necesario pasar tarjeta NFC para confirmar acción
Postcondición	Se crea un nuevo registro en la BD para un nuevo lote de Dureza y Conductividad

Caso de uso	Añadir Pieza A Bastidor/Lote
Actores	Operario
Resumen	Esta acción nos permite añadir una nueva pieza a un bastidor aun sin sellar para el proceso de Inspección por Líquidos Penetrantes o una nueva pieza para un lote en el caso del proceso de Dureza y Conductividad
Precondición	- Establecer conexión con la BD - Pieza disponible (que no haya sido ya sellada en otro bastidor o lote)
Postcondición	Se añade la pieza al bastidor o lote y se actualiza la BD con el nuevo registro

Caso de uso	Escanear tarjeta NFC
Actores	Operario
Resumen	Solicitará al operario escanear la tarjeta por el lector NFC
Postcondición	Confirmará si la tarjeta es válida en caso afirmativo realizará la acción solicitada
Dependencias	• Crear lote • Guardar Cambios • Sellar Bastidor

4.2.3. Diseño de la Base De Datos

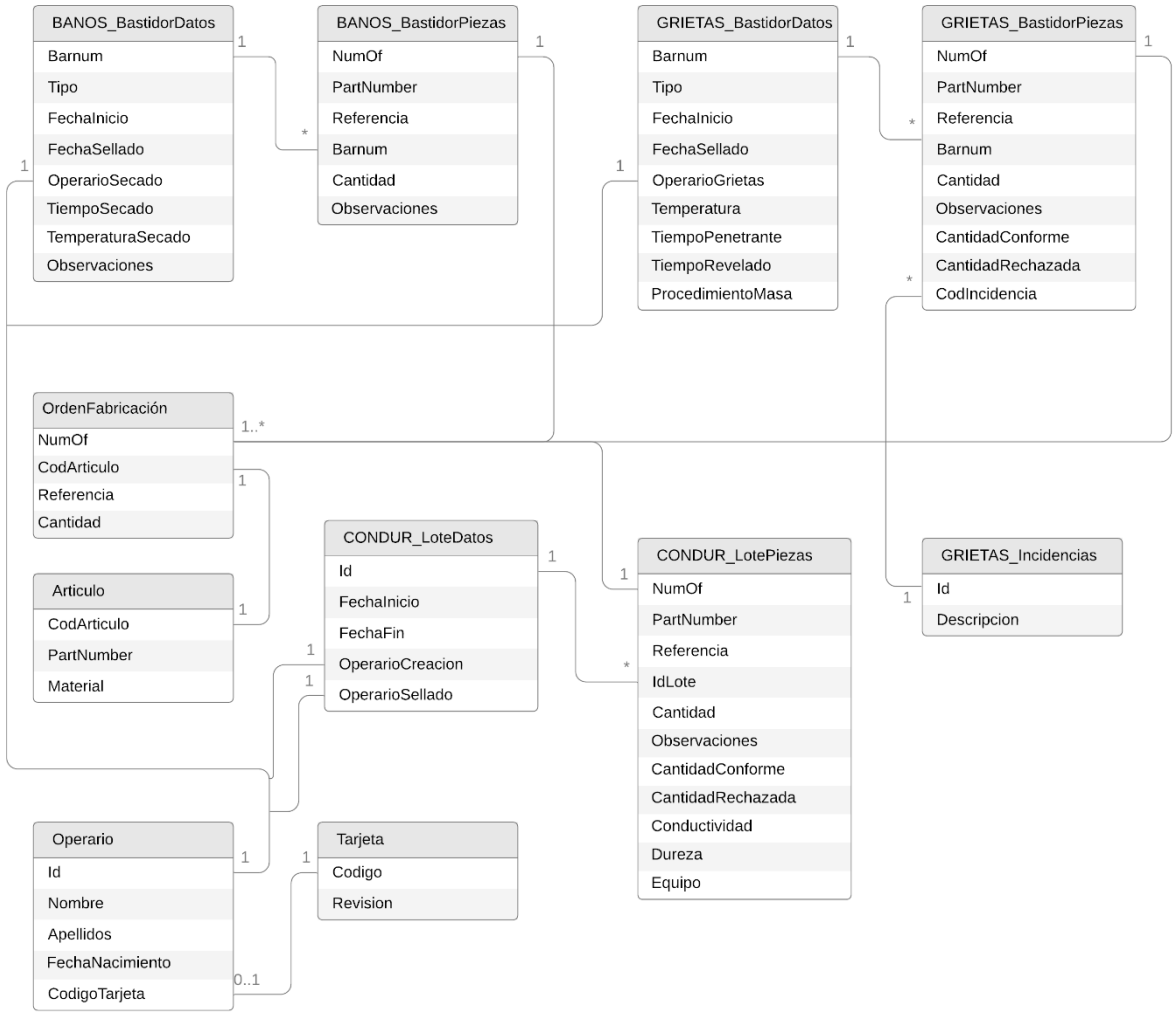


Figura 5 – Base de datos de la aplicación

Disponen de dos bases de datos una hecha en Access y otra en SQL Server.
La base de datos en Access sigue activa debido a que aún existen aplicaciones hechas en Visual Basic con Access que continúan haciendo uso de ella.
Además, también haremos uso de determinadas tablas de la Base de Datos en SQL Server para la realización de nuestra aplicación.

La base de datos hecha con SQL Server se usa para las nuevas aplicaciones que se han ido desarrollando y poco a poco se van migrando tablas de la base de datos de Access a esta nueva, con el objetivo de tener una única base de datos.

Para la realización de la base de datos de esta aplicación he tenido que migrar las tablas usadas en la Base de Datos desarrollada en Access a la nueva Base de Datos existente en Microsoft SQL Server. Solo ha sido necesario incluir aquellas tablas que utiliza la aplicación anterior a esta, ya que la base de datos contiene mucha más información de la que haremos uso, por lo que crearemos solo aquellas tablas que no existan en la nueva Base de Datos en SQL Server.

La migración de las tablas a la nueva base de datos en SQL Server supondrá una mejora del rendimiento y escalabilidad, seguridad mejorada, más usuarios simultáneos, mayor disponibilidad, entre otras mejoras.

El sistema permitirá a los operarios de la fábrica registrar y gestionar los resultados obtenidos de las pruebas realizadas durante los ensayos no destructivos en piezas de aviación.

La figura 5 muestra la base de datos de nuestra aplicación, la cual forma parte de la base de datos actual de la empresa hecha en SQL Server.

- La tabla BANOS_BastidorDatos se usa para consultar los bastidores que han finalizado todos los procesos de la sección de trabajo de Baños y además se registran los resultados obtenidos del proceso de Secado. Esta tabla pertenece a la sección de trabajo anterior al de Grietas, Baños.
  Barnum → Identificador único para cada bastidor
  OperarioSecado → Código identificador del operario responsable del proceso de Secado del Bastidor
  Datos que se registran tras el sellado: TiempoSecado, TemperaturaSecado, Observaciones, FechaSellado.

- La tabla BANOS_BastidorPiezas tiene un registro de todas las piezas de los bastidores de Baños. Esta tabla al igual que la anterior también pertenece al proceso anterior al de Grietas.
  NumOf →Identificador único por cada orden de fabricación
  Barnum → Número de bastidor al que ha sido asignado la orden

- En la tabla GRIETAS_BastidorDatos, el sistema registra los bastidores que son sellados en el proceso de Baños (cuando un bastidor es sellado en BAÑOS se añade al dato “OperarioSecado” el Operario responsable del sellado). De esta manera tenemos preparados los batidores para empezar el proceso de Inspección por Líquidos Penetrantes en nuestra aplicación.
  Barnum → Identificador único para cada bastidor
  OperarioGrietas → Código identificador del operario encargado de sellar el bastidor en el proceso de Inspección Por Líquidos Penetrantes
  Datos que se registran tras el sellado: Temperatura, TiempoPenetrante, TiempoRevelado, FecheSellado.

- La tabla GRIETAS_BastidorPiezas tiene un registro de todas las piezas de los bastidores de Grietas, los cuales son cargados por el sistema una vez hayan finalizado los procesos de Baños.
  NumOf →Identificador único para cada orden de fabricación
  Barnum → Número de bastidor al que ha sido asignado la orden
  CodIncidencia → Código identificador del número de incidencia

- La tabla GRIETAS_INCIDENCIAS contiene una lista con las incidencias más relevantes.
  Id → Identificador único para cada incidencia

- La tabla Articulo contiene registradas todas las piezas de aviación existentes.
  CodArticulo → Código Identificador único para cada pieza

- La tabla OrdenFabricacion registra los pedidos de los clientes (esta acción se realiza desde otra aplicación). Cuando un cliente realiza un pedido solicita las piezas y la cantidad que desea, por cada pieza solicitada se crea una "OF" (orden de fabricación) y se registra en la tabla. Posteriormente esa orden de fabricación es asignada a un bastidor para que se realicen las pruebas necesarias.
  NumOf → Identificador único por cada orden de fabricación
  CodArticulo → Código del articulo solicitado para la orden de fabricación

- La tabla CONDUR_LoteDatos registra los lotes que los operarios de la fábrica crean desde la aplicación para llevar a cabo el proceso de Dureza y Conductividad, se registra tanto el operario que creo el lote como el que la sello, ya que no siempre es el mismo.
  Id → Identificador único para cada lote
  OperarioCreacion → Código del operario que creo el lote
  OperarioSellado → Código del operario que sello el lote

- La tabla CONDUR_LotePiezas contiene un registro de las piezas de cada lote, las cuales son añadidas por los operarios durante el proceso de Dureza y Conductividad.
  NumOf → Identificador único por orden de fabricación
  IdLote → Id del lote al que ha sido asignado la orden

4.3. Desarrollo de la aplicación

4.3.1. Sprint 1 – Historial y Trabajo en Curso

Análisis

En este sprint se realizarán las siguientes tareas:

- Realización de una ventana principal que nos permita acceder a las distintas funcionalidades de la aplicación. Ver Figura 6.
- Aplicar funcionalidad a los botones para que nos muestren los históricos de cada proceso. Ver Figura 7.
- Botón y ventana para visualizar el de Trabajo en Curso. Ver Figura 10 y 12.
- Botón para actualizar la ventana de Trabajo en Curso. Ver Figura 10.

Diseño

Para la implementación de la ventana Principal haremos uso los controles Ribbon Control y XtraTabControls de Devexpress:

- Ribbon Control: Es una interfaz de usuario que permite agrupar los comandos de menú en páginas/pestañas y categorías. En comparación con las barras estándar y el modelo de menú, los Ribbon Control tienen las siguientes ventajas: Interfaz de usuario simplificada, acceso fácil y rápido de comandos, interfaz de usuario más compacta.
- Ribbon Page: Se representan como pestañas divididas estructural y visualmente en grupos que pueden mostrar varios comandos, ítems estáticos, editores y galerías.
- XtraTabControl: Es un contenedor de etiquetas de páginas (TabPages), que nos permite colocar nuestros controles.
- Ribbon status bar: La barra de estado diseñada para usarse junto con RibbonControl.

Figura 6 – Ventana Principal

Desarrollo

En esta parte explicaré el procedimiento que he seguido para implementar la funcionalidad de los botones que desarrollaremos en este sprint:

Figura 7 – Botones que muestran el histórico de los procesos

Ribbon Page “HISTORIAL”: Tiene 3 botones cada uno de ellos nos muestra el histórico relacionado con el proceso al que está asociado (Secado, Inspección Por Líquidos Penetrantes y Dureza y Conductividad).

Voy a explicar el procedimiento que he seguido para implementar el histórico de Secado del bastidor.

Bastidor	Tipo	Fecha Inicio	Fecha Sellado	Operario Grietas	Emulsion Temp.	Emulsion Tiempo	Alcalino Temp.
153292	Decapado	10/03/2020	10/03/2020	821	57,3	5	57,9
153291	Decapado	10/03/2020	10/03/2020	821	58,9	5	58,4
153289	Decapado	10/03/2020	10/03/2020	821	58,7	5	58,3
153287	Decapado	10/03/2020	10/03/2020	821	58,9	5	57,8
153285	Decapado	10/03/2020	10/03/2020	821	58,9	5	58,5
153283	Decapado	10/03/2020	10/03/2020	821	58,1	5	58,3
153282	Titanio	10/03/2020	10/03/2020	821	22,3	5	57,9

Figura 8 – Pestaña Historial Secado

Para el Secado he creado un control de usuario, una clase que hereda de XtraUserControl, esta clase de DevExpress nos permite crear un módulo independiente con controles y componentes que se pueden reutilizar en toda la aplicación. Para la visualización de la información he añadido al control de usuario un “GridControl”, este control nos permite mostrar la información de distintas formas, la que hemos elegido ha sido la vista cuadriculada (GridView), ya que nos interesa ver los datos como si de una tabla de Excel se tratase.

La información la obtenemos de una consulta SQL que hacemos a la base de datos a la tabla BANOS_BastidorDatos. Al trabajar con GridControl podemos insertar directamente la información obtenida de la consulta en forma de DataTable el cual luego introduciremos directamente en el DataSource del GridControl. Para la consulta he hecho uso de SqlCommand, SqlDataAdapter y SqlConnection para seleccionar registros de la base de datos y rellenar un DataTable con las filas obtenidas. A continuación, se devuelve el DataTable con los datos.

```
command = new SqlCommand(sSql, this._cn);

adapter = new SqlDataAdapter(command);

resultado = new DataTable("HistorialSecado");

adapter.Fill(resultado);
```

Figura 9 – Código que recupera datos y los guarda en un DataTable

El procedimiento seguido para los otros dos históricos es prácticamente el mismo, la diferencia en estos está en la consulta que hacemos a la base de datos.

Ribbon Page “TRABAJO EN CURSO”: Tiene dos botones, el primero (“Bastidores pendientes”) nos muestra los bastidores pendientes de los procesos de Secado e Inspección por Líquidos Penetrantes y el segundo (“Actualizar”) actualiza la información de los bastidores pendientes.

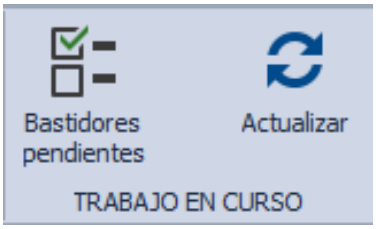


Figura 10 – Botones Trabajo En Curso

Para los Bastidores Pendientes he creado un control de usuario, una clase que hereda de XtraUserControl. He añadido al XtraUserControl un par de GridControls, uno para visualizar los bastidores pendientes de Secado y el otro para los de Inspección Por Líquidos Penetrantes.
El proceso para obtener las tablas como DataTable son similares a las realizadas para los históricos, la consulta SQL es lo que cambia. Recuperamos la información de las tablas BANOS_BastidorDatos, GRIETAS_BastidorDatos y CONDUR_LoteDatos
He añadido en la tabla una columna “Abrir” con un botón por cada línea de bastidor, con la idea de que al hacer clic abra el registro de datos de ese proceso en concreto. Para ello he tenido que hacer uso de RepositoryItemButtonEdit, que representa un repositorio de elementos que almacena configuraciones específicas para un control ButtonEdit, con esto conseguimos que todos los botones tengan las mismas características y realicen la acción deseada.

```
repositoryButtonSecado.Buttons[0].Kind = ButtonPredefines.Redo;
repositoryButtonSecado.Buttons[0].IsLeft = true;
repositoryButtonSecado.Buttons[0].Caption = "Editar";
repositoryButtonSecado.Buttons[0].Width = 238;
repositoryButtonSecado.Buttons[0].Appearance.BackColor2 = Color.Snow;
repositoryButtonSecado.Buttons[0].Appearance.BackColor = Color.Silver;
repositoryButtonSecado.ButtonClick += RepositoryItemButtonEditSecado_ButtonClick;
this.gridViewSecado.OptionsView.ShowButtonMode = DevExpress.XtraGrid.Views.Base.ShowButtonModeEnum.ShowAlways;
this.gridViewSecado.CustomRowCellEdit += gridViewSecado_CustomRowCellEdit;
```

Figura 11 – Propiedades usadas del RepositoryItemButtonEdit

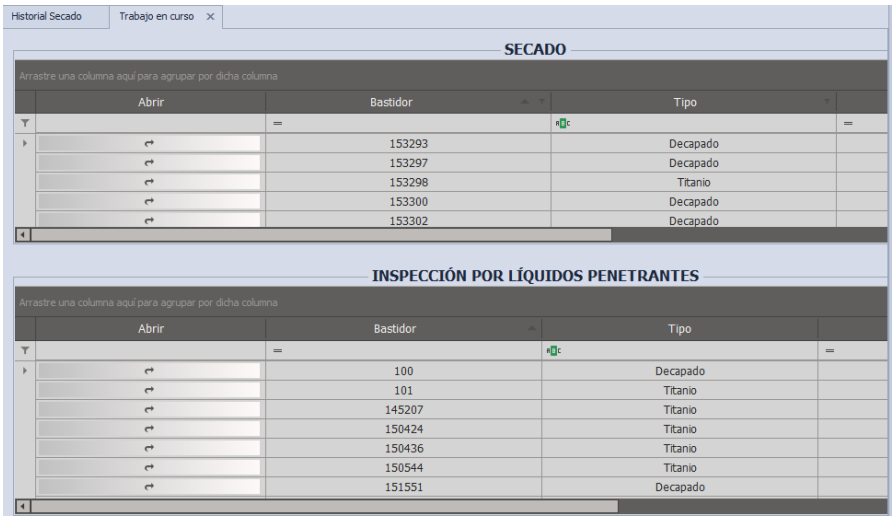


Figura 12 – Ventana de Trabajo En Curso

El botón Actualizar en un principio aparecerá deshabilitado, se activará cuando haya abierta una pestaña con los batidores pendientes (Trabajo en Curso) y se desactivará cuando cerremos esta pestaña. Este botón lo que hace es actualizar los datos mostrados de los bastidores pendientes.

Para ello empleamos un componente de Windows Forms llamado BackGroundWorker[1].

Lo que he hecho es modificar el evento RunWorkerCompleted[2] de este componente, el cual se inicia cuando hacemos clic en el botón Actualizar para que recargue la pestaña Bastidores Pendientes, de esta manera realiza una nueva consulta obteniendo los datos actualizados.

Utilizo un backgroundworker para que realice esta acción en segundo plano y así el resto del programa continúe funcionando, evitando que se cuelgue.

```
private void barButtonActualizar_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e)
{
    backgroundWorkerRecargarDatos.RunWorkerAsync();
    barButtonActualizar.Enabled = false;

    XtraTabPage pestanaACerrar = null;
    foreach (XtraTabPage page in this.xtraTabControlPrincipal.TabPages)
    {
        if (page.Name == PESTANA_TRABAJO_EN_CURSO)
            pestanaACerrar = page;
    }
    cerrarPestana(pestanaACerrar);
}

private void backgroundWorkerRecargarDatos_RunWorkerCompleted(object sender, System.ComponentModel.RunWorkerCo
{
    TrabajoEnCurso trabajo;
    if (!activarPestanaSiExiste(PESTANA_TRABAJO_EN_CURSO))
    {
        this.xtraTabControlPrincipal.TabPages.Add(PESTANA_TRABAJO_EN_CURSO);
        this.xtraTabControlPrincipal.TabPages.Last().Name = PESTANA_TRABAJO_EN_CURSO;
        this.xtraTabControlPrincipal.SelectedTabPageIndex = this.xtraTabControlPrincipal.TabPages.Count - 1;
        trabajo = new TrabajoEnCurso(this);
        trabajo.Name = PESTANA_TRABAJO_EN_CURSO;
        trabajo.Parent = this.xtraTabControlPrincipal.SelectedTabPage;
        trabajo.Location = new Point(5, 5);
        trabajo.Height = this.xtraTabControlPrincipal.Height - 40;
        trabajo.Dock = DockStyle.Fill;
    }
    barButtonActualizar.Enabled = true;
}
```

Figura 13 – Código empleado para el botón Actualizar

El código lo que hace es realizar en segundo plano todo lo que viene a continuación del componente backgroundWorkerRecargaDatos.RunWorkerAsync()[3], cerrando la pestaña bastidores pendientes. Una vez acaba esta acción ejecuta el código del evento RunWorkerCompleted, que lo hace es crear una nueva pestaña de Bastidores Pendientes con los datos actualizados haciendo una nueva consulta a la base de datos. Una vez acaba este evento la acción en segundo plano termina.

[1] Ejecuta una operación en un subproceso distinto
[2] Se produce cuando la operación en segundo plano se ha completado, se ha cancelado o ha producido una excepción
[3] Inicia la ejecución de una operación en segundo plano

Pruebas

Para la realización de las pruebas he comprobado el correcto funcionamiento ejecutando varios accesos a la aplicación.

Las pruebas han sido ejecutadas desde la interfaz del usuario para comprobar su correcto comportamiento, y han sido todas favorables. Las pestañas, botones y sus respectivos eventos no han presentado fallos. Se han realizado varias pruebas para la recuperación de información de la base datos y no se han encontrado fallos.

También hemos realizado múltiples pruebas para el botón de Actualizar del apartado de Trabajo En Curso, las cuales han sido todas satisfactorias. Se han probado varios casos donde podría dar problemas, por ejemplo, abrir la misma pestaña más de una vez, que se congele cuando se ejecuta en segundo plano, el bloqueado del botón durante su ejecución.

Revisión

Al terminar el sprint se habían realizado todas las tareas planificadas y la aplicación hasta el momento es totalmente funcional.

No se excedió el tiempo planificado, y me dispongo a realizar el siguiente sprint como se ha planificado en el cronograma.

4.3.2. Sprint 2 – Proceso Secado del Bastidor (Aluminio y Titanio)

Análisis

En este sprint se realizarán las siguientes tareas:

- Realización de las ventanas Decapado y Aluminio. Ver Figuras 14 y 15.
- Aplicar funcionalidad a los botones de cada ventana: Guardar, Cancelar, Sellar. Ver Figuras 21, 23 y 26.
- Añadir funcionalidad al botón Abrir para que permita acceder a estas nuevas ventanas (Decapado y Aluminio) creado en el Sprint 1. Ver Figura 17.

Diseño

Dado que el proceso seguido para la implementación de ambas ventanas es similar voy a explicar la llevada a cabo en la ventana de Aluminio. La diferencia más significativa entre ambas ventanas son los datos que se visualizan en ellas.

Como se muestra a continuación se puede observar que son ventanas muy similares y únicamente se diferencian por la información mostrada en el control de instalación y control de piezas.

Figura 14 – Ventana Aluminio

Figura 15 – Ventana Titanio

Para el desarrollo de la ventana de Aluminio, he creado un control de usuario que implementa los siguientes componentes de DevExpress:

- LayoutControl: permite organizar cualquier control dentro de un formulario sin que se superpongan y desalineen.
- GroupControl: es un control de contenedor que nos permite definir grupos de controles. Muestra un marco alrededor de los controles
- LayoutControlGroup: Un grupo regular con o sin encabezado y bordes.
- LabelControl: admite texto formateado, imágenes, cadenas de texto de varias líneas y formato HTML.

- TexEdit: Es un editor de texto simple. Es compatible con todas las funciones básicas de edición de texto, incluidas las selecciones y el menú contextual incorporado.
- MemoEdit: es un editor de texto desplegable.
- ComboBoxEdit: control es una lista desplegable que admite varios modos de operación.
- SimpleButton: Botón básico. Permite mostrar texto junto con una imagen personalizada y se puede hacer clic en tiempo de ejecución sin recibir foco.
- LayoutControlItem: muestra un control incrustado y una etiqueta opcional.
- EmptySpaceItem: no muestre nada: son 'celdas vacías' dentro de la tabla de diseño.
- GridControl: Muestra datos de una fuente de datos en varios formatos

Haciendo uso de todos estos componentes de DevExpress a través del Diseñador de Windows Forms conseguimos obtener un resultado como el mostrado a continuación:

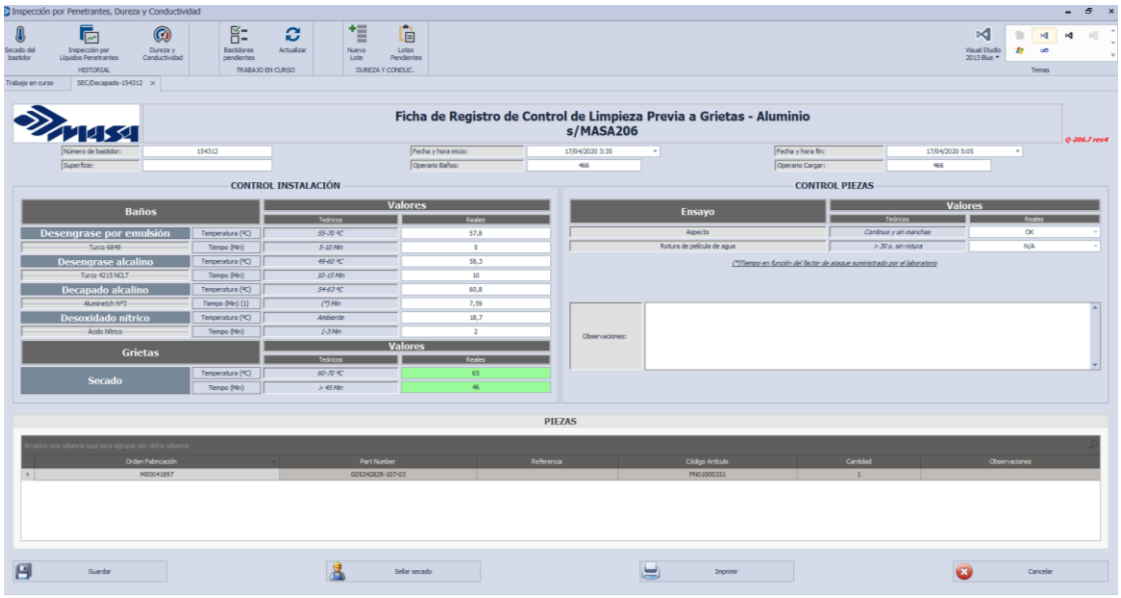


Figura 16 – Ventana Aluminio

Para añadir la funcionalidad a los botones de Abrir, he hecho lo siguiente:

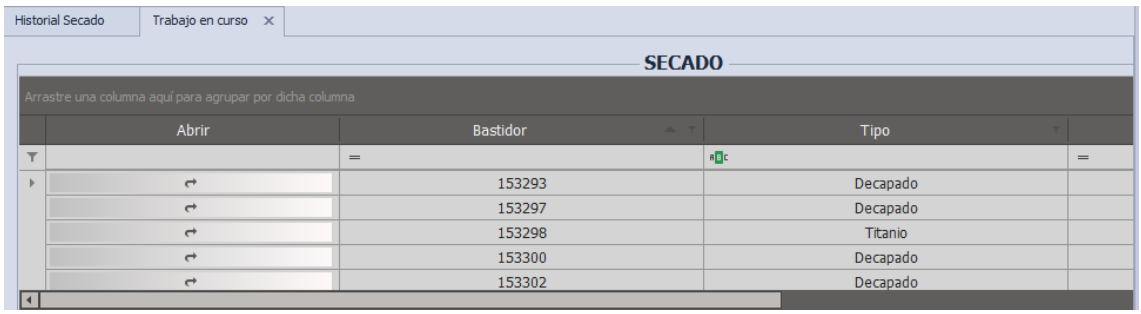


Figura 17 – Botón Abrir

La acción de este evento ocurre cuando se da clic en alguno de los botones Abrir, y el evento hace los siguiente:

- Primero recupera el bastidor y tipo de la fila del boton cliclado.
- Comprueba que no haya una pestaña del mismo abierta . Para ello llama al método activarPestanaSiExiste() del formulario Principal, al cual le pasa una cadena con el nombre de la pestaña seguido del bastidor y tipo. Este metodo lo que hace es recorrer todas las pestañas activas del formulario, de esta manera sabemos si hay una pestaña del mismo ya activa.

```
public bool activarPestanaSiExiste(string textoPestana)
{
    int i = 0;
    bool encontrado = false;

    while (!encontrado && i < xtraTabControlPrincipal.TabPages.Count)
    {
        if (xtraTabControlPrincipal.TabPages[i].Name.Equals(textoPestana))
        {
            xtraTabControlPrincipal.TabPages[i].PageVisible = true;
            xtraTabControlPrincipal.SelectedTabPage = xtraTabControlPrincipal.TabPages[i];
            encontrado = true;
        }
        i++;
    }
    return encontrado;
}
```

Figura 18 – Código para comprobar si la pestaña está activa

- En el caso de que no haya un pestaña activa con ese numero de bastidor, se añade a las pestañas del XtraTabControlPrincipal una pestaña con ese bastidor y tipo. Despues creamos la ventana con el tipo del bastidor, para ello declaramos una instancia del XtraUserControl

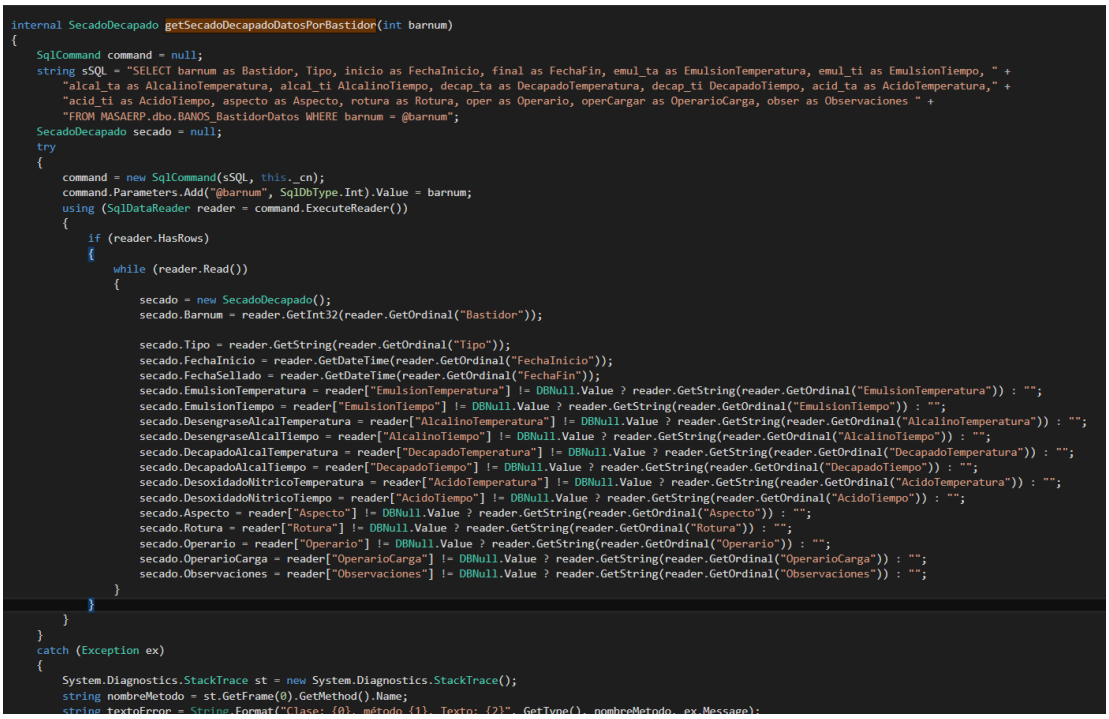
```
private void RepositoryItemButtonEditSecado_ButtonClick(object sender,ButtonPressedEventArgs e)
{
    int bastidor = (int)gridViewSecado.GetRowCellValue(gridViewSecado.FocusedRowHandle,"Bastidor");
    string tipo = (string)gridViewSecado.GetRowCellValue(gridViewSecado.FocusedRowHandle,"Tipo");

    if (!xtraFromPrincipal.activarPestanaSiExiste(PESTANA_SECADO + "/" + tipo + "-" + bastidor))
    {
        xtraFromPrincipal.xtraTabControlPrincipal.TabPages.Add(PESTANA_SECADO + "/" + tipo + "-" + bastidor);
        xtraFromPrincipal.xtraTabControlPrincipal.TabPages.Last().Name = PESTANA_SECADO + "/" + tipo + "-" + bastidor;
        xtraFromPrincipal.xtraTabControlPrincipal.SelectedTabPageIndex = xtraFromPrincipal.xtraTabControlPrincipal.TabPages.C
        if (tipo == "Decapado")
        {
            Decapado decapado = new Decapado(bastidor,xtraFromPrincipal.xtraTabControlPrincipal);
            decapado.Name = PESTANA_SECADO + "/" + tipo + "-" + bastidor;
            decapado.Parent = xtraFromPrincipal.xtraTabControlPrincipal.SelectedTabPage;
            decapado.Location = new Point(5,5);
            decapado.Height = xtraFromPrincipal.xtraTabControlPrincipal.Height - 40;
            decapado.Dock = DockStyle.Fill;

        }
        else if (tipo == "Titanio")
        {
            Titanio titanio = new Titanio(bastidor,xtraFromPrincipal.xtraTabControlPrincipal);
            titanio.Name = PESTANA_SECADO + "/" + tipo + "-" + bastidor;
            titanio.Parent = xtraFromPrincipal.xtraTabControlPrincipal.SelectedTabPage;
            titanio.Location = new Point(5,5);
            titanio.Height = xtraFromPrincipal.xtraTabControlPrincipal.Height - 40;
            titanio.Dock = DockStyle.Fill;
        }
```

Figura 19 – Código empleado en el evento clic del RepositoryItemButtonEdit

- Por ultimo en el método Load() del XtraUserControl recuperamos los datos del bastidor, para después llamar al método getDecapadoPorBastidor() de la Lógica de Negocio, el cual recupera la información del bastidor haciendo una consulta a la base de datos a través de la Persistencia de nuestro Sistema.

Todo este proceso lo repetimos tanto para el Secado (Decapado y Aluminio) como Inspeccion Por Líquidos Penetrantes y Dureza Y Conductiidad. Recuperamos los datos de las tablas dependiendo del tipo de bastidor:

- Secado: tablas BANOS_BastidorDatos y BANOS_BastidorPiezas
- Inspeccion por Líquidos Penetrantes: GRIETAS_BastidorDatos y GRIETAS_BastidorPiezas
- Dureza y Conductividad: CONDUR_LoteDatos y CONDUR_LotePiezas



```
internal SecadoDecapado getSecadoDecapadoDatosPorBastidor(int barnum)
{
    SqlCommand command = null;
    string sSQL = "SELECT barnum as Bastidor, Tipo, inicio as FechaInicio, final as FechaFin, emul_ta as EmulsionTemperatura, emul_ti as EmulsionTiempo, " +
        "alcal_ta as AlcalinoTemperatura, alcal_ti AlcalinoTiempo, decap_ta as DecapadoTemperatura, decap_ti DecapadoTiempo, acid_ta as AcidoTemperatura," +
        "acid_ti as AcidoTiempo, aspecto as Aspecto, rotura as Rotura, oper as Operario, operCargar as OperarioCarga, obser as Observaciones " +
        "FROM MASAERP.dbo.BANOS_BastidorDatos WHERE barnum = @barnum";
    SecadoDecapado secado = null;
    try
    {
        command = new SqlCommand(sSQL, this._cn);
        command.Parameters.Add("@barnum", SqlDbType.Int).Value = barnum;
        using (SqlDataReader reader = command.ExecuteReader())
        {
            if (reader.HasRows)
            {
                while (reader.Read())
                {
                    secado = new SecadoDecapado();
                    secado.Barnum = reader.GetInt32(reader.GetOrdinal("Bastidor"));

                    secado.Tipo = reader.GetString(reader.GetOrdinal("Tipo"));
                    secado.FechaInicio = reader.GetDateTime(reader.GetOrdinal("FechaInicio"));
                    secado.FechaSellado = reader.GetDateTime(reader.GetOrdinal("FechaFin"));
                    secado.EmulsionTemperatura = reader["EmulsionTemperatura"] != DBNull.Value ? reader.GetString(reader.GetOrdinal("EmulsionTemperatura")) : "";
                    secado.EmulsionTiempo = reader["EmulsionTiempo"] != DBNull.Value ? reader.GetString(reader.GetOrdinal("EmulsionTiempo")) : "";
                    secado.DesengraseAlcalTemperatura = reader["AlcalinoTemperatura"] != DBNull.Value ? reader.GetString(reader.GetOrdinal("AlcalinoTemperatura")) : "";
                    secado.DesengraseAlcalTiempo = reader["AlcalinoTiempo"] != DBNull.Value ? reader.GetString(reader.GetOrdinal("AlcalinoTiempo")) : "";
                    secado.DecapadoAlcalTemperatura = reader["DecapadoTemperatura"] != DBNull.Value ? reader.GetString(reader.GetOrdinal("DecapadoTemperatura")) : "";
                    secado.DecapadoAlcalTiempo = reader["DecapadoTiempo"] != DBNull.Value ? reader.GetString(reader.GetOrdinal("DecapadoTiempo")) : "";
                    secado.DesoxidadoNitricoTemperatura = reader["AcidoTemperatura"] != DBNull.Value ? reader.GetString(reader.GetOrdinal("AcidoTemperatura")) : "";
                    secado.DesoxidadoNitricoTiempo = reader["AcidoTiempo"] != DBNull.Value ? reader.GetString(reader.GetOrdinal("AcidoTiempo")) : "";
                    secado.Aspecto = reader["Aspecto"] != DBNull.Value ? reader.GetString(reader.GetOrdinal("Aspecto")) : "";
                    secado.Rotura = reader["Rotura"] != DBNull.Value ? reader.GetString(reader.GetOrdinal("Rotura")) : "";
                    secado.Operario = reader["Operario"] != DBNull.Value ? reader.GetString(reader.GetOrdinal("Operario")) : "";
                    secado.OperarioCarga = reader["OperarioCarga"] != DBNull.Value ? reader.GetString(reader.GetOrdinal("OperarioCarga")) : "";
                    secado.Observaciones = reader["Observaciones"] != DBNull.Value ? reader.GetString(reader.GetOrdinal("Observaciones")) : "";
                }
            }
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.StackTrace st = new System.Diagnostics.StackTrace();
        string nombreMetodo = st.GetFrame(0).GetMethod().Name;
        string textoError = String.Format("Clase: {0}, método {1}, Texto: {2}", GetType(), nombreMetodo, ex.Message);
```

Figura 20 – Código, ejemplo de consulta para recuperar los datos del Secado de tipo Decapado

Para añadir funcionalidad a los botones he adaptado los eventos clic de cada botón:

- Botón Cancelar:

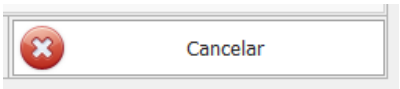


Figura 21 – Botón Cancelar

Modificamos el evento click del botón para cancelar la edición del bastidor y además también controlamos que no se pierda información en caso de que el operario olvide guardar los cambios efectuados.

```
private void btnCancelar_Click_1(object sender,EventArgs e)
{
    if (this._MODIFICADO)
    {
        if (XtraMessageBox.Show("Se han detectado modificaciones en los campos,
        {
            XtraTabPage page = this._xtraTabControlPrincipal.SelectedTabPage;
            this._xtraTabControlPrincipal.TabPages.Remove(page);
        }
    }
    else
    {
        XtraTabPage page = this._xtraTabControlPrincipal.SelectedTabPage;
        this._xtraTabControlPrincipal.TabPages.Remove(page);
    }
}
```

Figura 22 – Código empleado en el evento clic del Botón Cancelar

- Botón Guardar:

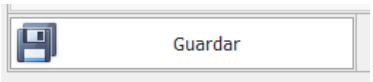


Figura 23 – Botón guardar

Modificamos el evento clic del botón, para llamar a un método de la Lógica de Negocio el cual conecta con la Persistencia y de esta manera realizamos un Update de los datos del bastidor con el que se está trabajando. Con este evento actualizamos toda información excepto los operarios de sellado, ya que estos son registrados en el evento del botón Sellar.

```
private void btnGuardar_Click(object sender, EventArgs e)
{
    _logicaNeg.guardarDatosSecado(_bastidor, txtSecTemp.Text, txtSecTi.Text, txtObserv.Text);
}
```

Figura 24 – Código empleado en el evento clic del Botón Guardar

```
internal bool guardarDatosSecado(int bastidor, string secadoTemperatura, string secadoTiempo, string observaciones)
{
    string sSql = "UPDATE MASAERP.dbo.BANOS_BastidorDatos SET secad_ta = @SecadoTemperatura, secad_ti = @SecadoTiempo, obser = @Observaciones WHERE barnum = @Bastidor";
    bool update = false;
    try
    {
        using (SqlCommand command = new SqlCommand(sSql,_cn))
        {
            command.Parameters.Add("@SecadoTemperatura",SqlDbType.Int).Value =  secadoTemperatura;
            command.Parameters.Add("@SecadoTiempo",SqlDbType.NVarChar).Value = secadoTiempo;
            command.Parameters.Add("@barnum",SqlDbType.NVarChar).Value =  bastidor;
            command.Parameters.Add("@Observaciones",SqlDbType.NVarChar).Value = observaciones;


            int rows = command.ExecuteNonQuery();
            if (rows == 1)
                update = true;
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.StackTrace st = new System.Diagnostics.StackTrace();
        string nombreMetodo = st.GetFrame(0).GetMethod().Name;
        string textoError = String.Format("Clase: {0}, método {1}. Texto: {2}",GetType(),nombreMetodo,ex.Message);
        ClaseError regError = new ClaseError(GetType().Assembly.GetName().Name,DateTime.Now,textoError,Environment.UserName);
        RegistrarSuceso.registrarUnError(regError);
        throw;
    }
    return update;
}
```

Figura 25 – Código, ejemplo de Update de la tabla de Secado para el guardado

- Botón Sellar:

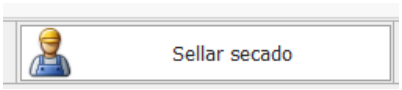


Figura 26 – Botón Sellar

Modificamos el evento click del botón, para llamar a un método de la Lógica de Negocio el cual conecta con la Persistencia y de esta manera realizamos un Update de los datos del bastidor con el que se está trabajando. En este evento se actualiza toda la información del bastidor incluido los operaríos de sellado.

```
private void btnSellar_Click(object sender, EventArgs e)
{
    if (txtSecTi.BackColor == Color.PaleGreen && txtSecTemp.BackColor == Color.PaleGreen) {
        _logicaNeg.sellarBastidorSecado(_bastidor, txtSecTemp.Text, txtSecTi.Text, txtObserv.Text, t
    }
}
```

Figura 27 – Código empleado en el evento clic del Botón Sellar

```
}

internal bool sellarBastidorSecado(int bastidor, string secadoTemperatura, string secadoTiempo, string observaciones, string operario)
{
    string sSql = "UPDATE MASAERP.dbo.BANOS_BastidorDatos SET secad_ta = @SecadoTemperatura, secad_ti = @SecadoTiempo, obser = @Observaci
    bool update = false;
    try
    {
        using (SqlCommand command = new SqlCommand(sSql,_cn))
        {
            command.Parameters.Add("@SecadoTemperatura",SqlDbType.Int).Value =  secadoTemperatura;
            command.Parameters.Add("@SecadoTiempo",SqlDbType.NVarChar).Value = secadoTiempo;
            command.Parameters.Add("@barnum",SqlDbType.NVarChar).Value =  bastidor;
            command.Parameters.Add("@Observaciones",SqlDbType.NVarChar).Value = observaciones;
            command.Parameters.Add("@Operario", SqlDbType.NVarChar).Value = operario;

            int rows = command.ExecuteNonQuery();
            if (rows == 1)
                update = true;
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.StackTrace st = new System.Diagnostics.StackTrace();
        string nombreMetodo = st.GetFrame(0).GetMethod().Name;
        string textoError = String.Format("Clase: {0}, método {1}. Texto: {2}",GetType(),nombreMetodo,ex.Message);
        ClaseError regError = new ClaseError(GetType().Assembly.GetName().Name,DateTime.Now,textoError,Environment.UserName);
        RegistrarSuceso.registrarUnError(regError);
        throw;
    }
    return update;
}
```

Figura 28 – Código, ejemplo de Update de la tabla de Secado para el sellado

Cada vez que se edite algún dato de la cabecera o de los controles de Instalación y Piezas, se mostrará un aviso en caso de que el operario cierre la pestaña o de clic en el botón de cancelar sin haber guardado los cambios antes. El aviso alertará a los operarios que los cambios que han realizado no han sido guardados, de esta manera evitamos la perdida de información.

En las clases Decapado, Titanio, Inspección Por Líquidos Penetrantes y Dureza y Conductividad tenemos una variable booleana que nos indica si se ha realizado algún cambio en los datos del bastidor seleccionado, es decir, si el operario realiza algún cambio en los datos del bastidor esta variable pasará a tener valor true. De esta manera, tenemos un control del cierre de la ventana. Para ello hemos editado el evento CloseButtonClick[1] del control Principal, de forma comprueba el estado de

[1] Se produce cuando se hace clic en el botón Cerrar que se muestra dentro del encabezado de XtraTabControl.

la variable (de la clase de la pestaña) cada vez que se cierre una pestaña de las mencionadas.

El estado de la variable lo controlamos a través de dos eventos:

- EditValueChanged[1] en el caso de los textEdit, ComboBoxEdit, etc
- CellValueChanging[2] en el caso de los GridView)

La siguiente imagen muestra el código que controla el cierre de la pestaña en caso de no haber guardado los cambios:

```
private void xtraTabControlPrincipal_CloseButtonClick(object sender, EventArgs e)
{
    XtraTabPage pestanaACerrar = (e as DevExpress.XtraTab.ViewInfo.ClosePageButtonEventArgs).Page as XtraTa
    bool cerrar = true;

    foreach (Control control in pestanaACerrar.Controls)
    {
        if (control is UserControl)
        {
            if (control.Name.StartsWith("SEC"))
            {
                if (control.Name.Split('/')[1].StartsWith("Decapado"))
                {
                    Decapado decapado = (Decapado) control;
                    if (decapado._MODIFICADO == true)
                        cerrar = false;
                }
```

Figura 29 – Parte de código del evento CloseButtonClick, ejemplo en el caso del Decapado

Pruebas

Para la realización de las pruebas he comprobado el correcto funcionamiento ejecutando varios accesos a la aplicación.

Las pruebas han sido ejecutadas desde la interfaz del usuario para comprobar su correcto comportamiento, y han sido todas favorables. Hemos comprobado que los componentes de las ventanas de Decapado y Aluminio estén colocados correctamente y que sus respectivos eventos se ejecuten sin errores.

También hemos realizado múltiples pruebas para los botones de las ventanas (Cancelar, Guardar y Sellar), las cuales han sido todas satisfactorias. Se han comprobado que los eventos se ejecutan correctamente, las consultas realizadas no presentasen problemas, los casos de perdida de información (botón Cancelar o cierre de la pestaña) estén controlados.

Para la funcionalidad del botón Abrir de la ventana de Trabajo en Curso hemos probado varios casos donde podría dar problemas, por ejemplo, que solo se puede

[1] Evento que Se activa inmediatamente después de que se haya cambiado el valor de edición.
[2] Evento que se activa cuando se modifica un valor de la celda, se escribe o elimina un carácter, elige un valor de la lista desplegable, etc.

abrir una pestaña por bastidor, que permita abrir y cerrar las veces necesarias, que se ejecute la consulta correctamente.

Revisión

Al terminar el sprint se habían realizado todas las tareas planificadas y la aplicación hasta el momento es totalmente funcional.

No se excedió el tiempo planificado, y me dispongo a realizar el siguiente sprint como se ha planificado en el cronograma.

4.3.3. Sprint 3 – Proceso Inspección por Líquidos Penetrantes

Análisis

En este sprint se realizarán las siguientes tareas:

- Realización de la ventana de Inspección por Líquidos Penetrantes. Ver Figura 28.
- Registrar la información de las piezas de cada bastidor
- Aplicar funcionalidad a los botones de cada ventana: Guardar, Cancelar, Sellar.

Diseño

Para el desarrollo de la ventana de Inspección por Líquidos Penetrantes he creado un control de usuario que implementa los mismos componentes que emplee para las ventanas de Decapado y Aluminio, siendo estos:

- LayoutControl
- GroupControl
- LayoutControlGroup
- LabelControl
- TexEdit
- MemoEdit
- ComboBoxEdit
- SimpleButton
- LayoutControlItem
- EmptySpaceItem
- GridControl

Empleando estos componentes de DevExpress por medio del Diseñador de Windows Forms obtenemos un resultado como el mostrado a continuación:

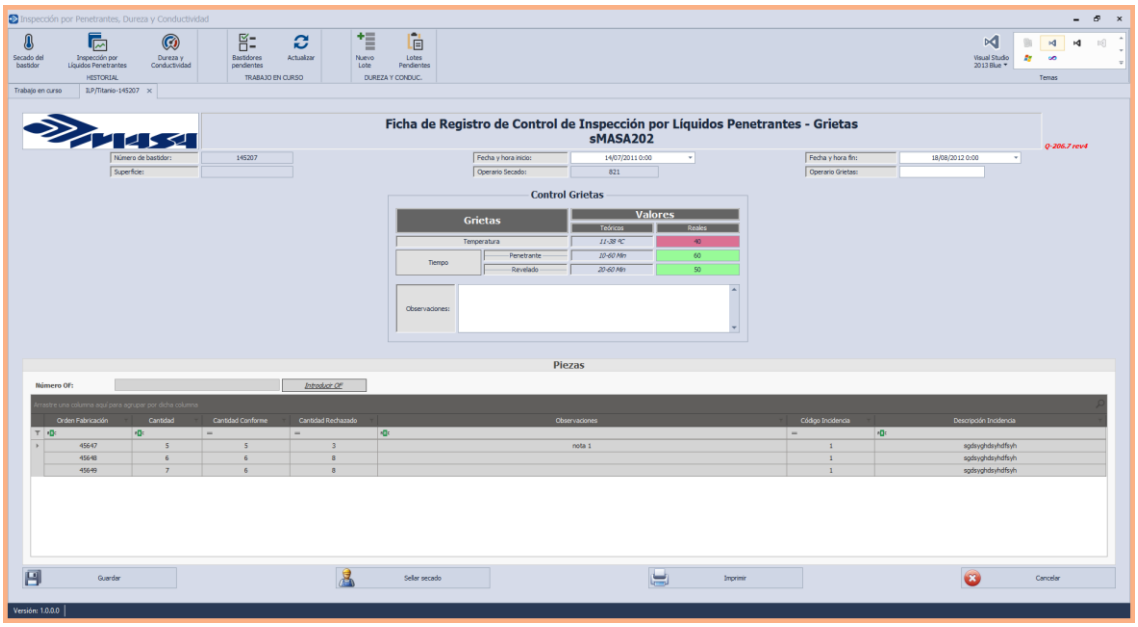


Figura 30 – Ventana Inspección Por Líquidos Penetrantes

Para la edición de datos de las piezas de cada bastidor hemos habilitado la configuración de las celdas del GridView de la ventana. De esta manera cada vez que sea editada se guardara el estado de esta con los datos actualizados en el DataSource del GridControl que contiene al GridView. Para ello, empleamos el evento CellValueChanging, que se activa cada vez que se modifica un valor del GridView.

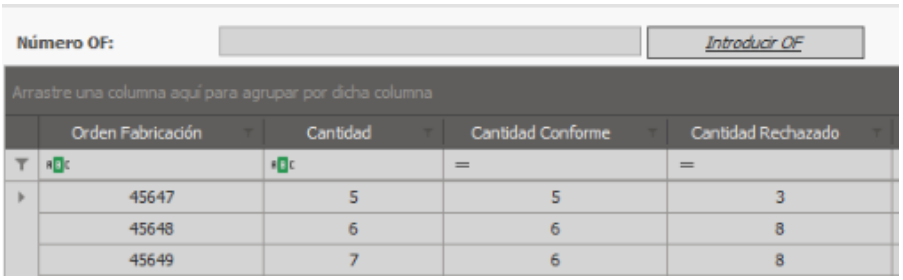


Figura 31 - Datos de las Piezas del Bastidor contenidas en un GridView

Con este método, los operarios pueden editar en cualquier momento cualquier dato de las piezas del bastidor de forma sencilla como si se tratase de una hoja Excel. Cada vez que se edita el GridView de las piezas, la cabecera de datos o el Control de Grietas se lanzará un mensaje de guardado pendiente en caso de que el operario cierre el bastidor sin haber realizado un guardado previo, para evitar la pérdida de información.

Para añadir una pieza al bastidor, editamos el evento Click del botón. De esta manera el operario podrá incluir nuevas piezas al bastidor, la pieza se añade a la tabla (al GridView del control) y DataSource del GridView.

La funcionalidad que hemos aplicado a los eventos clic de los botones Cancelar, Sellar y Guardar en estas ventanas son similares a las explicadas en el Sprint 2.

Pruebas

Para la realización de las pruebas he comprobado el correcto funcionamiento ejecutando varios accesos a la aplicación.

Las pruebas han sido ejecutadas desde la interfaz del usuario para comprobar su correcto comportamiento, y han sido todas favorables. Hemos comprobado que los componentes de la ventana de Inspección por Líquidos Penetrantes estén colocados correctamente y que sus respectivos eventos se ejecuten sin errores. Además, hemos realizado varias pruebas en las que añadimos piezas a distintos bastidores, las cuales no han presentado fallos.

También hemos realizado múltiples pruebas para los botones de las ventanas (Cancelar, Guardar y Sellar), las cuales han sido todas satisfactorias. Se han comprobado que los eventos se ejecutan correctamente, las consultas realizadas no presentasen problemas, los casos de perdida de información (botón Cancelar o cierre de la pestaña) estén controlados.

Revisión

Al terminar el sprint se habían realizado todas las tareas planificadas y la aplicación hasta el momento es totalmente funcional.

No se excedió el tiempo planificado, y me dispongo a realizar el siguiente sprint como se ha planificado en el cronograma.

4.3.4. Sprint 4 – Proceso Dureza y Conductividad

Análisis

En este sprint se realizarán las siguientes tareas:

- Realización de la ventana de Dureza y Conductividad (Nuevo Lote). Ver Figura 30.
- Añadir piezas (por OF) al bastidor.
- Realización de la ventana Lotes Pendientes.
- Aplicar funcionalidad a los botones de cada ventana: Guardar, Cancelar, Sellar.

Diseño

Para el desarrollo de la ventana de Dureza y Conductividad he creado un control de usuario que implementa los siguientes componentes:

- LayoutControl
- GroupControl
- LayoutControlGroup
- LabelControl
- TexEdit
- MemoEdit

- ComboBoxEdit
- SimpleButton
- LayoutControlItem
- EmptySpaceItem
- GridControl

Empleando estos componentes de DevExpress por medio del Diseñador de Windows Forms obtenemos un resultado como el mostrado a continuación:

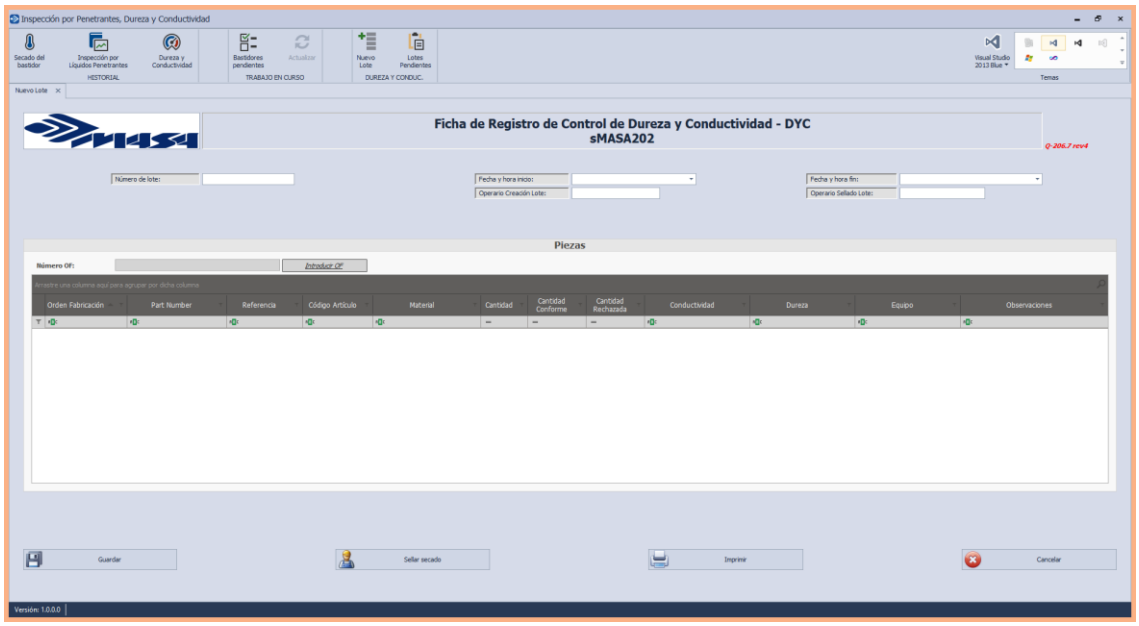


Figura 32 – Ventana Dureza Y Conductividad

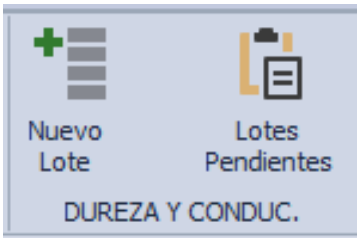


Figura 33 – Botones Dureza Y Conductividad

En el evento clic del botón Nuevo Lote, creamos una instancia del control de usuario de Dureza y Conductividad, es decir, añadimos al control Principal una nueva pestaña con un lote nuevo y de tipo DYC.

La funcionalidad que hemos aplicado a los eventos clic de los botones Cancelar, Sellar en estas ventanas son similares a las explicadas en el Sprint 2.

En el evento click del botón Guardar, hacemos lo siguiente:

- Si se trata de un nuevo lote, creamos un nuevo registro (Insert) en la tabla CONDUC_LotesDatos de la BD y además se registran las piezas, que han sido añadidas al lote, en la tabla CONDUR_LotesPiezas.

  Todo esto lo controlamos en un método de la Persistencia que llamamos desde la Lógica de Negocio. En el método creamos un objeto SqlTransaction llamando a BeginTransaction en el objeto SqlConnection. Todas las operaciones posteriores asociadas a la transacción (en este caso, la inserción de datos en las tablas CONDUR_LotesDatos y CONDUR_LotesPiezas) se realizan en el objeto de SqlTransaction.

  En el caso de que todas las consultas se realicen con éxito, se hará commit[1] de la transacción. Si falla alguna de las consultas se realizará un rollback[2] de la transacción.
  Usamos Try/Catch, el control de errores se utiliza para controlar los errores que se producen al intentar confirmar o revertir la transacción. Tanto Commit como Rollback generan un InvalidOperationException si se termina la conexión o si la transacción ya se ha revertido en el servidor.

```
internal bool crearNuevoLoteDurezaYConductividad(DYCDatos dycDatos)
{
    string sSql = "INSERT INTO MASAERP.dbo.CONDUR_LoteDatos (FechaIncio, OperarioCreacion, FechaFi
    bool insert = false;
    SqlTransaction trans = this._cn.BeginTransaction("insertar_Lote_transaction");
    SqlCommand command;
    int filasModificadas = 0;
    int filasAInsertar = dycDatos.Piezas.Count;
    try{
        command = new SqlCommand(sSql, this._cn, trans);
        command.Parameters.Add("@FechaInicio", SqlDbType.DateTime).Value = dycDatos.FechaInicio;
        Más Parámetros

        int rows = command.ExecuteNonQuery();
        command = new SqlCommand("SELECT @@IDENTITY", this._cn, trans);
        int idLote = Convert.ToInt32(command.ExecuteScalar());
        dycDatos.Id = idLote;

        if (idLote > 0){
            if (dycDatos.Piezas.Count > 0){
                sSql = "INSERT INTO MASAERP.dbo.CONDUR_LotePiezas(idLote, NumOf, Cantidad, Observa
                            "VALUES(@IdLote, @NumOf, @Cantidad, @Observaciones, @CantidadConforme
                command.CommandText = sSql;
                command.Parameters.Clear();
                command.Parameters.Add("@IdLote", SqlDbType.Int);
                Más Parámetros

                foreach (Pieza pieza in dycDatos.Piezas.Values){
                    command.Parameters["@IdLote"].Value = dycDatos.Id;
                    Más Parámatros
                    filasModificadas += command.ExecuteNonQuery();
                }
            }
        }
```

Figura 34 – Código para registrar un nuevo Lote en la tabla de Dureza Y Conductividad

[1] Confirma la transacción de base de datos.
[2] Revierte una transacción desde un estado pendiente.

- Si se trata de un lote ya registrado, actualizamos los datos del lote y de sus respectivas piezas (realizando un Update) en las tablas CONDUR_LoteDatos y CONDUR_LotePiezas.

  Al igual que el caso anterior, este método también lo controlamos desde la Persistencia, pero en este caso actualizamos (Update) los datos del lote en la tabla CONDUR_LoteDatos y para las piezas primero realizamos un borrado (Delete) en la tabla CONDUR_LotePiesas de las piezas del lote correspondiente para posteriormente añadir (Insert) las piezas del GridView del lote. Todas las operaciones posteriores asociadas a la transacción (Update, Delete e Insert) se realizan en el objeto de SqlTransaction.

  En el caso de que todas las operaciones se realicen con éxito, se hará commit de la transacción. Si falla alguna de las operaciones se realizará un rollback de la transacción.

  Usamos Try/Catch, el control de errores se utiliza para controlar los errores que se producen al intentar confirmar o revertir la transacción. Tanto Commit como Rollback generan un InvalidOperationException si se termina la conexión o si la transacción ya se ha revertido en el servidor.

```
internal bool actualizarLoteDurezaYConductividad(DYCDatos dycDatos)
{
    string sSql = "UPDATE MASAERP.dbo.CONDUR_LoteDatos SET FechaFin = @FechaFin WHERE id = @IdLote";
    bool update = false;
    SqlTransaction trans = this._cn.BeginTransaction("insertar_Lote_transaction");
    SqlCommand command;
    int filasModificadas = 0;
    int filasAInsertar = dycDatos.Piezas.Count;

    try{
        command = new SqlCommand(sSql, this._cn, trans);
        command.Parameters.Add("@FechaFin", SqlDbType.DateTime).Value = dycDatos.FechaFin;
        command.Parameters.Add("@IdLote", SqlDbType.Int).Value = dycDatos.Id;

        command.ExecuteNonQuery();

        sSql = "DELETE FROM MASAERP.dbo.CONDUR_LotePiezas WHERE idLote = @IdLote";

        command.CommandText = sSql;
        command.Parameters.Clear();
        command.Parameters.Add("@IdLote", SqlDbType.Int).Value = dycDatos.Id;

        command.ExecuteNonQuery();

        if (dycDatos.Piezas.Count > 0){
            sSql = "INSERT INTO MASAERP.dbo.CONDUR_LotePiezas(idLote, NumOf, Cantidad, Observaciones,
                   "VALUES(@IdLote, @NumOf, @Cantidad, @Observaciones, @CantidadConforme, @Cant
            command.CommandText = sSql;
            command.Parameters.Clear();

            command.Parameters.Add("@IdLote", SqlDbType.Int);
            Más Parámetros

            foreach (Pieza pieza in dycDatos.Piezas.Values){
                command.Parameters["@IdLote"].Value = dycDatos.Id;
                Más Parámetros
                filasModificadas += command.ExecuteNonQuery();
```

Figura 35 – Código para actualizar un Lote de la tabla de Dureza Y Conductividad

Pruebas

Para la realización de las pruebas he comprobado el correcto funcionamiento ejecutando varios accesos a la aplicación.

Las pruebas han sido ejecutadas desde la interfaz del usuario para comprobar su correcto comportamiento, y han sido todas favorables. Hemos comprobado que los componentes de la ventana de Inspección por Dureza y Conductividad estén colocados correctamente y que sus respectivos eventos se ejecuten sin errores. Además, hemos realizado varias pruebas en las que añadimos piezas (por OF) a distintos lotes, las cuales no han presentado fallos.

También hemos realizado múltiples pruebas para los botones de las ventanas (Cancelar, Guardar y Sellar), las cuales han sido todas satisfactorias. Se han comprobado que los eventos se ejecutan correctamente, las consultas realizadas no presentasen problemas, los casos de perdida de información (botón Cancelar o cierre de la pestaña) estén controlados.

Revisión

La realización de este sprint se vio interrumpida por la situación de la Pandemia del Covid-19, por lo que no se pudo completar el sprint en su totalidad. Quedo pendiente de realizar la ventana de Lotes Pendientes para dejar finalizado este sprint.

Debido a esta nueva situación no se pudo continuar con la realización de la aplicación, por que este sprint queda pendiente, así como el siguiente.

5. Seguimiento y Control

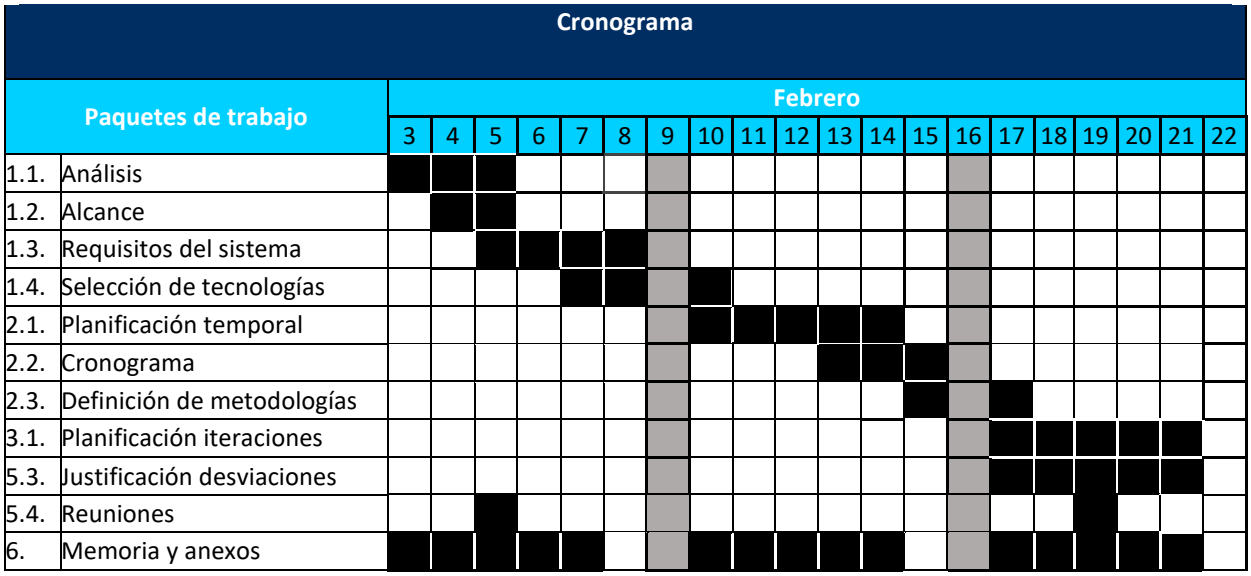
En este apartado veremos cómo se ajusta lo planificado a los hechos que realmente se han producido.

5.1. Duración real de las tareas

TAREA	INICIO	FIN	HORAS TOTALES
1. Definición del sistema	**3 de febrero**	**4 de febrero**	**13 horas**
1.1. Análisis	3 de febrero	5 de febrero	3 horas
1.2. Alcance	4 de febrero	5 de febrero	4 horas
1.3. Requisitos del sistema	5 de febrero	8 de febrero	2 horas
1.4. Selección de tecnologías	7 de febrero	10 de febrero	4 horas
2. Planificación	**10 de febrero**	**17 de febrero**	**19 horas**
2.1. Planificación temporal	10 de febrero	14 de febrero	10 horas
2.2. Cronograma	13 de febrero	15 de febrero	4 horas
2.3. Definición de metodologías	15 de febrero	17 de febrero	5 horas

3. Diseño e Implementación	**17 de febrero**	**8 de abril**	**193 horas**
3.1. Planificación iteraciones	17 de febrero	21 de febrero	8 horas
3.2. Diseño	24 de febrero	8 de abril	15 horas
3.3. Desarrollo de la aplicación	24 de febrero	8 de abril	170 horas
4. Pruebas	**28 de febrero**	**8 de abril**	**20 horas**
4.1. Pruebas SW	28 de febrero	8 de abril	20 horas
4.2. Pruebas tarjetas NFC	20 de marzo	8 de abril	0 horas
5. Seguimiento y control	**3 de febrero**	**3 de junio**	**35 horas**
5.1. Duración real de las tareas	3 de febrero	3 de junio	10 horas
5.2. Cronograma real	5 de febrero	3 de junio	7 horas
5.3. Justificación de desviaciones	17 de febrero	3 de junio	12 horas
5.4. Reuniones	3 de febrero	3 de junio	6 horas
6. Memoria y anexos	**5 de febrero**	**3 de junio**	**55 horas**

5.2. Cronograma real



Cronograma																				
Paquetes de trabajo		Febrero							Marzo											
		23	24	25	26	27	28	29	1	2	3	4	5	6	7	8	9	10	11	12	13
3.2.	Diseño y Diagramas UML																				
3.3.	Desarrollo del sistema																				
4.1.	Pruebas Software																				
5.3.	Justificación desviaciones																				
5.4.	Reuniones																				
6.	Memoria y anexos																				

Cronograma																					
Paquetes de trabajo		Marzo																		Abril	
		14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2
3.2.	Diseño y Diagramas UML																				
3.3.	Desarrollo del sistema																				
4.1.	Pruebas software																				
4.2.	Pruebas Tarjetas NFC																				
5.3.	Justificación desviaciones																				
5.4.	Reuniones																				
6.	Memoria y anexos																				

Cronograma																					
Paquetes de trabajo		Abril																			
		3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
3.3.	Desarrollo del sistema																				
5.1.	Pruebas Software																				
5.3.	Justificación desviaciones																				
5.4.	Reuniones																				
6.	Memoria y anexos																				

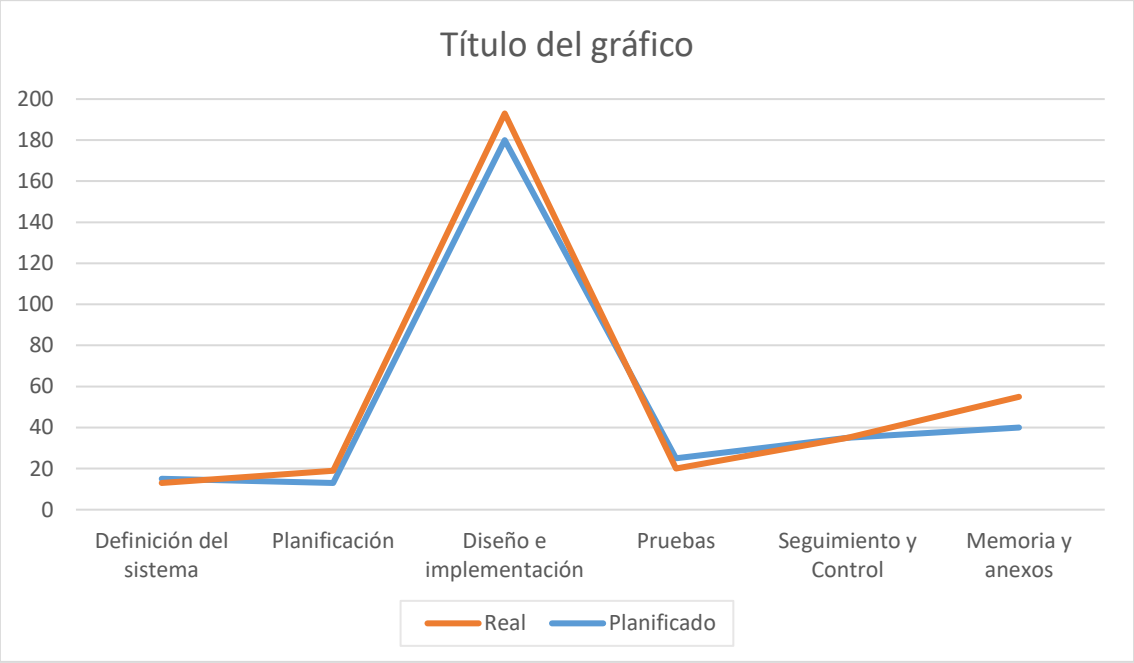
Cronograma																					
Paquetes de trabajo		Abril								Mayo											
		23	24	25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12
5.1.	Pruebas Software																				
5.2.	Pruebas tarjetas NFC																				
5.3.	Justificación desviaciones																				
5.4.	Reuniones																				
6.	Memoria y anexos																				

Cronograma																					
Paquetes de trabajo		Mayo																			
		13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1
5.3.	Justificación desviaciones																				
5.4.	Reuniones																				
6.	Memoria y anexos																				

Cronograma																					
Paquetes de trabajo		Junio																			
		2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
5.3.	Justificación desviaciones																				
5.4.	Reuniones																				
6.	Memoria y anexos																				

5.3. Justificación de desviaciones

En este apartado explicaré cómo difieren los tiempos reales de los tiempos planificados y el porqué.



En la figura tenemos una gráfica comparativa de tiempos fraccionados por fases principales de la planificación del proyecto. Podemos observar, que la curva de la planificación temporal queda por debajo de la curva que representa la realidad (hasta la fase de Pruebas).

Debido a la falta de experiencia en la gestión de proyectos, el inicio del proyecto requirió más tiempo del estimado, por lo que puede que la distribución temporal para las fases iniciales del proyecto se haya subestimado.

Como se puede observar en la gráfica, en cuanto a la fase de Diseño e implementación, la parte de implementación del software se ha alargado más de lo previsto. Esto se debe a que se han dedicado más horas al día de los estimados para la realización de esta fase. A pesar de que faltan por desarrollar algunas pocas funcionalidades del sistema, debido al parón por la pandemia, en términos generales la aplicación se encuentra en la fase final de su desarrollo, a falta de implementar parte del sprint 4 y el desarrollo del sprint 5 (tarjetas NFC).

Finalmente, si nos fijamos en la cola final de la curva (fase de Seguimiento y Control) podemos observar una leve sobrestimación temporal. En cuanto al tiempo estimado para la fase de Memoria y Anexos, se ha subestimado el tiempo que se ha planificado emplear para el desarrollo de esta parte. De nuevo, esto se debe a la inexperiencia en el ámbito de redacción de documentos de proyectos de este tipo.

6. Conclusiones

6.1. Objetivos alcanzados

Debido a la situación actual que estamos viviendo en España y resto del mundo, en la empresa también nos vimos afectados por lo que las prácticas se tuvieron que parar. Por todo esto, no se puedo completar en su totalidad la todas las funcionalidades que nos habíamos propuesto para la aplicación.

Entre los objetivos que nos marcamos en un principio, estaba el de realizar un sistema que permitiese controlar y gestionar los ensayos no destructivos en piezas de aviación. A pesar de las circunstancias, se han cumplido en gran parte las funcionalidades principales de la aplicación. La funcionalidad para gestionar los procesos de Secado e Inspección Por Penetrantes se han completado con éxito y están operativos para su uso. En el caso del proceso de Dureza y Conductividad, queda pendiente de realizar la ventana de Lotes Pendientes ya que no se pudo realizar su implementación por la interrupción del confinamiento.

6.2. Trabajo futuro

Este proyecto surgió de la idea de sustituir el sistema actual en Access. El objetivo principal era realizar una interfaz para el usuario que permita registrar los ensayos no destructivos sobre piezas de aviación. El sistema realizado está preparado para llevar a cabo los dos primeros procesos (Secado del Bastidor e Inspección Por Líquidos Penetrantes).

Debido a la interrupción por la pandemia, el proceso de Dureza y Conductividad quedo incompleto, por lo que se plantea como trabajo futuro a realizar cuando se reanuden las actividades.
Por otro lado, también quedo pendiente la implementación de las tarjetas NFC a nuestro sistema, la realización de este se llevará a cabo cuando las actividades en la empresa se reanuden.

Por todo esto, el trabajo futuro para este proyecto es la finalización del mismo, con el objetivo de ponerlo a disposición de los operarios una vez este completado.

7. Bibliografía

Recurso	Fuente	Disponible en
Componentes DevExpress	DevExpress	https://www.devexpress.com/
GridViews	DevExpress	https://docs.devexpress.com/WindowsForms/3464/controls-and-libraries/data-grid/views/grid-view
RepositoryItem ButtonEdit	DevExpress	https://docs.devexpress.com/WindowsForms/DevExpress.XtraEditors.Repository.RepositoryItemButtonEdit.Buttons
C#	MSDN	https://docs.microsoft.com/es-es/dotnet/csharp/
Transacciones	MSDN	https://docs.microsoft.com/es-es/dotnet/api/system.data.sqlclient.sqlconnection.begintransaction?view=dotnet-plat-ext-3.1
Comparativa y consultas	Stack Overflow	https://stackoverflow.com/
Datos Empresa	MASA	https://masa.aero/